

AD-A121 794

ADVANCED AVIONIC SYSTEMS FOR MULTIMISSION APPLICATIONS  
VOLUME II(U) BOEING MILITARY AIRPLANE CO SEATTLE WA  
L A SMITH ET AL. OCT 82 AFWAL-TR-82-1076-VOL-2  
F33615-77-C-1252

1/1

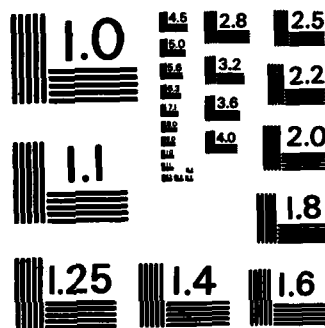
UNCLASSIFIED

F/G 9/2

NL

END

FILMED  
\*  
DTIC



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

AD A121794

AFWAL-TR-82-1076  
Volume II



ADVANCED AVIONIC SYSTEMS FOR MULTIMISSION APPLICATIONS

Boeing Military Airplane Company  
Seattle, Washington 98124

October 1982

NOV 26 1982

Final Report for Period May 1979 - June 1980

A

Approved for public release, distribution unlimited.

AVIONICS LABORATORY  
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES  
AIR FORCE SYSTEMS COMMAND  
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

82 11 26 010

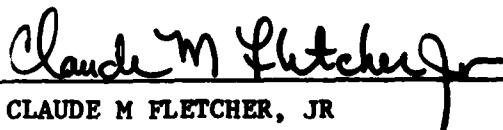
DTIC FILE COPY

NOTICE

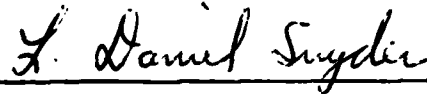
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

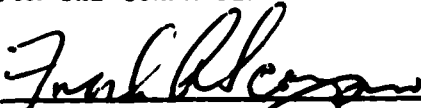


CLAUDE M FLETCHER, JR  
PROJECT ENGINEER  
Mission Software & System  
Integration Group



L DANIEL SNYDER, Chief  
Mission Software & System Integration Group  
System Avionics Division

FOR THE COMMANDER



FRANK A SCARPINO  
Acting Chief, System Avionics Division  
Avionics Laboratory

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/AAAS-1, W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFWAL-TR-82-1076, Volume II	2. GOVT ACCESSION NO. AD-A121 794	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ADVANCED AVIONIC SYSTEMS FOR MULTIMISSION APPLICATIONS		5. TYPE OF REPORT & PERIOD COVERED Final Report for Period May 79 - Jun 80
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Leroy A. Smith, Stephen W. Behnen, Keith D. Pratt, Mack B. McCall, et al.		8. CONTRACT OR GRANT NUMBER(s) F33615-77-C-1252
9. PERFORMING ORGANIZATION NAME AND ADDRESS Boeing Military Airplane Company Seattle, Washington 98124		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project 2003, Task 01, Work Unit 10
11. CONTROLLING OFFICE NAME AND ADDRESS Avionics Laboratory (AFWAL/AAAS-1) Air Force Wright Aeronautical Laboratories (AFSC) Wright-Patterson Air Force Base, Ohio 45433		12. REPORT DATE October 1982
		13. NUMBER OF PAGES 81
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)  Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Portions of this report were presented at the 1980 NAECON conference in a paper entitled, "A Single Processor Synchronous Executive Derived from the DAIS Executive," and authored by S. W. Behnen and R. L. Gutmann. The computer data contained in this technical report are theoretical and in no way reflect Air-Force-		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)  Computers, avionics, synchronous systems, DAIS program, software executive, microprocessors, information transfer systems, hardware standards, software standards		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)  This study produced system control procedures and executive software design specifications for three different information transfer systems (ITS), each designed to implement multimission aspects of an avionic system. The stationary master is the best understood ITS and has multimission advantages if the applications software is designed for change. The non-stationary master is an excellent candidate for a pod-oriented multimission		

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

Block 20 (continued)

- application. The contention access ITS is designed to be most flexible in terms of change, at the potential cost of higher initial integration checkout due to the asynchronous nature of the communication.
- A second task was to design, develop and build a compact version of the DAIS executive that would function in a one processor system and support only synchronous bus communications. This executive, called the Single Processor Synchronous Executive (SPSE), was tested and delivered to AFWAL.

The primary goals of this task were to build a functional executive that:

1. ➤ Maintains the DAIS executive-to-applications interface;
  2. ➤ Communicates on a MIL-STD-1553A bus;
  3. ➤ Is coded in J73/I;
  4. ➤ Supports the avionic system load for an AMST or modern tactical fighter aircraft;
  5. ➤ Uses DAIS support software (LINKS, ALAP, PALEFAC, PALEFAC processor); and
  6. ➤ Requires substantially less memory than the baseline DAIS executive.
- All goals were achieved.

Block 18 (continued)

owned software programs.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

## FOREWORD

### VOLUME II

This final technical report for the Advanced Avionic Systems for Multi-Mission Applications (AASMMMA) was prepared by The Boeing Military Airplane Co. (BMAC), Seattle, Washington. The final report consists of three separately bound volumes which covers the work performed under contract F33615-77-C-1252 during the period of January 1978 to June 1981.

The program was performed in two phases for the Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, Ohio 45433. The first phase covered three tasks which addressed (1) Distributed Avionics Information System Design, (2) Avionic Cost Analysis Methods & Models, and (3) Embedded Microcomputer Standardization Concepts. These tasks were conducted for AFWAL/AAAA. The contract monitor was Mr Gary Wambold, the program manager was Mr Donald E Dewey, and the principal investigators were Dr Leroy A Smith and Mr Al Crossgrove. Volume I of this report describes this phase.

The second phase of the program Volumes II and III covered tasks which addressed (1) the Development & Evaluation of Advanced Digital Avionics System Architectures and (2) the Development of a Single Processor Synchronous Executive (SPSE) derived from the Digital Avionics Information System (DAIS) Executive. These tasks were conducted for AFWAL/AAAS and the AFWAL contract manager was Mr Claude M Fletcher, Jr, the Boeing program manager was Dr Leroy A Smith, and the principal investigator was Mr Stephen W Behnen.

The other significant contributors to this effort included:

Richard F Bousley  
Tammy R Cremeen  
Dr Robert L Gutmann  
John J Henrick  
James H Mason  
Mack B McCall  
Kevin M McMahon

Michael E McSharry  
Keith D Pratt  
Gerald Sommerman  
Laura Townsend  
Frank E Troth  
C Ray Turner

all from The Boeing Company; James Gracia, Edward Comer, and Joseph Malnar, all from the Harris Corporation; Capt Robert Percefull from the U S Air Force; and Lynn Trainor from the Systran Corporation.



Approved For  
Distribution  
Excluded From  
Automatic  
Downgrading  
and  
Declassification

A

## TABLE OF CONTENTS

	<u>Page</u>
1.0 Introduction . . . . .	1
1.1 Scope . . . . .	1
1.2 Background . . . . .	1
2.0 AASMMA Program Summary - Tasks IV and V. . . . .	3
3.0 Task Description Summary . . . . .	5
3.1 Task IV - Summary of the Information Transfer Systems. . . . .	5
3.1.1 Stationary Master ITS - SMITS. . . . .	5
3.1.2 Nonstationary Master ITS . . . . .	5
3.1.3 Contention Multiple Access ITS . . . . .	6
3.1.4 Comparison of Information Transfer Systems . . . . .	7
3.1.5 Task IV Conclusions. . . . .	7
3.2 Task V - SPSE Summary. . . . .	8
4.0 Detailed Program Findings. . . . .	11
4.1 Candidate Information Transfer Systems for Multimission Applications . . . . .	11
4.1.1 Stationary Master. . . . .	11
4.1.1.1 SMITS Control Procedures . . . . .	11
4.1.1.2 SMITS Executive Functional Description . . . . .	13
4.1.2 Nonstationary Master . . . . .	15
4.1.2.1 NSMITS Control Procedures. . . . .	16
4.1.2.2 NSMITS Executive Design. . . . .	17
4.1.3 Contention Multiple Access Information Transfer System . . . . .	18
4.1.3.1 CMAITS Control Procedures. . . . .	19
4.1.3.2 CMAITS Executive Design. . . . .	20
4.2 Analysis of Information Transfer System Performance. . . . .	21
4.2.1 Message Wait Time. . . . .	21
4.2.1.1 Stationary Master ITS (SMITS). . . . .	21
4.2.1.2 Nonstationary Master ITS (NSMITS). . . . .	27
4.2.1.3 Contention Multiple Access ITS (CMAITS). . . . .	28
4.2.2 Information Throughput . . . . .	29
4.2.2.1 Stationary Master ITS (SMITS). . . . .	29



## TABLE OF CONTENTS (Concluded)

	<u>Page</u>
4.2.2.2 Nonstationary Master ITS (NSMITS) . . . . .	29
4.2.2.3 Contention Multiple Access ITS (CMAITS) . . . . .	31
4.3 Single Processor Synchronous Executive . . . . .	32
4.3.1 Task Purpose . . . . .	32
4.3.2 Task Approach . . . . .	32
4.3.3 Documentation Produced . . . . .	33
4.3.4 Support Tools . . . . .	33
4.3.4.1 Potential Development Problems . . . . .	35
4.3.4.2 SUBSORT . . . . .	36
4.3.4.3 Universal Source Files . . . . .	38
4.3.4.4 INGEUNS . . . . .	38
4.3.5 Description of SPSE . . . . .	40
4.3.6 Comparison of DAIS Executive and SPSE . . . . .	40
4.3.7 Modifications to DAIS Executive Structure . . . . .	42
4.3.8 Modifications to the Data Base Structure . . . . .	42
4.3.9 Demonstration Results . . . . .	43
4.3.10 SPSE Performance . . . . .	45
5.0 Recommendations . . . . .	47
<u>APPENDICES</u>	
A Rationale for Stationary Master Information Transfer System Control Procedures . . . . .	49
B Rationale for Nonstationary Master Information Transfer System Control Procedures . . . . .	64
C Rationale for Contention Multiple-access Information Transfer System Control Procedures . . . . .	75

## LIST OF ILLUSTRATIONS

	<u>Page</u>
Figure 2-1 Master Program Schedule . . . . .	4
Figure 4-1 Effect of Number of Devices on Message Latency . . . . .	23
Figure 4-2 Effect of Message Length on Message Latency . . . . .	24
Figure 4-3 Effect of Bandwidth on Message Latency . . . . .	25
Figure 4-4 Effect of Trigger Message on Contention Message Latency . . .	26
Figure 4-5 Effect of Bandwidth on Throughput . . . . .	30
Figure 4-6 SUBSORT Inputs and Outputs . . . . .	37
Figure 4-7 Generating Individual Source Files from a USF . . . . .	39
Figure 4-8 SPSE Memory Requirement Versus Time . . . . .	46

## LIST OF TABLES

Table 3-1 COMPARISON OF DAIS EXECUTIVE AND SPSE. . . . .	10
Table 4-1 TASK V DOCUMENTATION . . . . .	34
Table 4-2 PMD DELETIONS IN SPSE. . . . .	44

## DEFINITIONS AND ACRONYMS

AASMMA	Advanced Avionic Systems for Multi-Mission Applications
ADA	New DoD Language Replacing J73
AFWAL	Air Force Wright Aeronautical Laboratories
ALAP	Avionics Laboratory Assembler Program
BIU	Bus Interface Unit
BMU	Bus Monitor Unit
BMAC	Boeing Military Airplane Company
CDR	Critical Design Review
CMAITS	Contention Multiple Access Information Transfer System
CPU	Central Processing Unit
DAIS	Digital Avionic Information System
DARTS	Distributed Architecture Research Test System
DOD	Department of Defense
HOL	High Order Language
INGEUNS	INTERpreter for GEnerating Executives from UNiversal Source files
ITS	Information Transfer System
LINKS	Software Test Stand Linker
MAP	Model Avionics Program
MMU	Mass Memory Unit
NSMITS	Nonstationary Master Information Transfer System
OFF	Operational Flight Program
PALEFAC	Partitioning, Analyzing, Linking, Editing Facility
PCP	Processor Control Panel
PDR	Preliminary Design Review
PE	Processing Element
PMD	PALEFAC Mission Data
RAM	Random Access Memory
ROM	Read Only Memory
RT	Remote Terminal
SCADU	Super Control and Display unit
SCP	System Control Procedures
SIL	Synchronous Instruction List
SMITS	Stationary Master Information Transfer System
SPSE	Single Processor Synchronous Executive
URT	Universal Remote Terminal
USF	Universal Source File
V&V	Validation and Verification

## 1.0 INTRODUCTION

### 1.1 SCOPE

This technical report summarizes the activities and results of the Advanced Avionic Systems for Multi-Mission Applications (AASMMMA) program. Phase 1 of the AASMMMA program studied current and projected information transfer system designs and architectures for avionic systems which require a multi-mission capability. Volume I of this report summarizes the Phase 1 activities.

The purpose of Phase 2 was: (1) to examine in depth and to document the information transfer systems derived in phase 1; (2) to design, develop, and deliver a compact version of the DAIS executive which functions in a one processor system and which supports only synchronous bus communications. Volume 2 summarizes phase 2, which is documented in detail by: control procedures and part one executive specifications for each information transfer system; and, parts one and two executive specifications, control procedures, a second AASMMMA interim report, and test plans, procedures and reports for the single processor synchronous executive. The background of phase 2 of the AASMMMA program is summarized in paragraph 1.2. Paragraph 2.0 describes the purpose of the tasks in phase 2. Paragraph 3.0 is a summary of the work performed and paragraph 4.0 expands the summary by describing the detailed work documented. Recommendations for further work are provided in paragraph 5.0.

### 1.2 BACKGROUND

In the past few years it has been the goal of the Air Force to develop and apply methods and technologies that would permit avionic systems evolve in an orderly manner as mission needs change. A lack of interface commonality among avionic systems has made the task of system design and integration, as well as the task of upgrading or modifying systems, very costly. The Digital Avionics Information System (DAIS) concept was established by the Air Force to investigate and establish standard interfaces among the various elements of avionic systems to reduce this cost. These concepts are being matured and are being considered for several near term retrofit airplane programs (F-111 A&E, F-4, F-15, F-16) and are planned for future systems.

The concept of multimission roles for a single airframe (or a restricted family of airframes) is influencing our military weapon planning. Threats, which are changing more rapidly than ever before, make it necessary to plan for mission-adaptive and threat-adaptive avionic suites over the life of an airframe. Two multimission concepts are emerging. One approach is to design a "core" set of avionics and separable "peripheral" avionics so that the avionics suite can be readily changed by removing and replacing mission dependent functions. Another approach is to depend on well established interface standards (e.g., standard hardware and software modules) which permit an avionics system to be updated (retrofit) throughout the life of the airframe. These approaches are not mutually exclusive, and they can be complementary. Therefore, the adaptation of current interface standards and the exploitation of new digital technologies to achieve the multimission capability is required.

Multimission roles can be readily accomplished if the multimission functions can be isolated and made independent. Isolation (independence) between functions can be achieved by using the inherent separation of functions found in hierarchical architectures. Such architectural features would make it easier to develop, integrate, maintain and modify (update) an avionics system or to use it in an aircraft having multimission roles. Most avionics technology and design standards, including the DAIS related standards, were designed primarily for single level architectures using minicomputers as the processing elements. However, multilevel or hierarchical architectures require a high degree of distributed processing. This factor alone has precluded the use of this architecture because of the high cost of military minicomputers. Now that microprocessor technology has progressed to the point where military qualified microcomputers cost substantially less than military qualified minicomputers, the hierarchical architecture using distributed processing can become a reality.

## 2.0 AASMMA PROGRAM SUMMARY - TASKS IV AND V

Phase 2 of the AASMMA program consisted of Tasks IV and V. The purpose of Task IV was to further examine the three candidate information transfer systems defined in Task I. The result of this effort was the generation of a system control procedure and a part one computer program design specification for each system. The purpose of Task V was to design, develop, build, and deliver a compact version of the DAIS executive which would function in a one processor system and support only synchronous bus communications. This task was started in June 1979 and was completed in April 1980. The master program schedule for AASMMA is shown in Figure 2-1.

Program element	FY78			FY79			FY80					
	CY1978			CY1979			CY1980					
	J	F	M	A	M	J	J	F	M	A	M	J
Task I (candidate ITS development and technology forecast)												
Task II (cost model selection/development)												
Task III (HW and SW standards study)												
Task IV (ITS detailed definition)												
Task V (single processor synchronous executive)												
Final report												

Figure 2-1. Master Program Schedule

### 3.0 TASK DESCRIPTION SUMMARY

Tasks IV and V of the AASMMA program were concerned with the development of: the detailed description of three ITS designs, including system control procedures and Part I computer program design specifications (Task IV); a single processor synchronous executive member of the DAIS executive family (Task V).

#### 3.1 TASK IV - SUMMARY OF INFORMATION TRANSFER SYSTEMS

Three separate information transfer systems (ITS) were defined in Phase I to support multimission applications. These are the stationary master, nonstationary master and contention access ITS. The common ground for the development of each of these ITS is that each is based on the system defined in the DAIS program and each retains the DAIS executive/applications interface. Therefore, applications software designed to operate with one ITS should run with any of the information transfer systems.

##### 3.1.1 Stationary Master ITS - SMITS

The stationary master ITS is the closest of the three ITS to the current DAIS system. The DAIS system is composed of a set of one to four minicomputers (AN/AYK-15) communicating among each other and among remote terminals via a dual redundant serial bus (MIL-STD-1553A/DAIS). The executive in the AN/AYK-15 is designed to support applications software task communication to such an extent that individual tasks could be moved from one AN/AYK-15 minicomputer to another simply by altering the executive communication tables. The concepts embodied in the DAIS program have proven to be an appropriate baseline upon which to build and expand to accommodate the microprocessor and more highly integrated avionic systems of the 1980's and especially to build low life cycle cost avionic systems that are adaptable to growth and change.

The enhancements made to the DAIS ITS for the stationary master ITS include three specific areas: multibus capability, distributed processing accommodation, and MIL-STD-1553B bus protocol.

##### 3.1.2 Nonstationary Master ITS

The nonstationary master concept was developed to have an independent controller for each mission or sub-mission function. A system configuration would consist of a controller for the core avionics which is used on every mission and a separate controller for each separate pod- or pallet-mounted set of equipment. There will probably be very few controllers (on the order of two or three), so a simple round-robin control mechanism was chosen. Other than the control transfer and some device management, the functioning of the nonstationary master and stationary master is identical. The description of the stationary master applies



equally well to the nonstationary master. The processing element at the start of the cycle is the primary master and the other controllers are called secondary masters. During the time that any master has control, it acts as a stationary master controlling all transactions on the bus. The primary master has the additional duties of initiating all minor cycles and monitoring each secondary master so that it does not exceed its allotted time for control. The primary master is also responsible for reconfiguration due to failure of any secondary masters. One secondary master is designated as a monitor for the primary master in case of primary failure. Each master may be responsible for the control of devices on the bus, but the spheres of control will be mutually exclusive, so that one device will not be controlled by more than a single master even though several masters might communicate with a given device. The primary master will be responsible for the control of the majority of devices. Each secondary master will probably be an interbus processor serving some specialized subsystem such as that associated with pod-mounted equipment, while the primary master controls the core avionic systems.

### 3.1.3 Contention Multiple Access ITS

The contention multiple access information transfer system is considerably different in terms of bus control from the other two ITS. In the contention multiple access information transfer system, there are no unique bus masters, rather each device contends for use of the bus. When a device wishes to transmit, it will sense the communication activity on the bus media to determine if a transmission is allowed. If activity is detected, the device will wait until the transmission ceases, then the device will begin a random delay associated with its queued message's priority before attempting to transmit. If the bus has remained free of communication activity for the duration of the delay, the device initiates transmission. Utilizing this approach it is possible for multiple devices to attempt communications on the bus simultaneously. If this occurs, a collision is said to have occurred, and the transmissions are terminated and rescheduled for transmission later. Once a device acquires the bus and begins communication which does not result in a collision, it may transmit until a given length of time expires. During this period of time as many complete contiguous messages can be transmitted in the time interval will be scheduled and transmitted. This contention mechanism allows considerable flexibility in system integration. The advantage of easily providing for new, modified or multi-mission oriented sensors is obvious because of the independent nature of the processors. Modifications to the system will primarily affect the processors responsible for control of the modified section and others will remain unaffected. Each of the messages are addressed by content rather than by origin or destination and multiple bus interface units can accept the same messages because of the broadcast nature of the protocol. The operation of such a transmission system is primarily asynchronous in nature. Any cyclic operations are scheduled to be performed by the executive using the processing element's local clock. This local synchronism, coupled with time tagging of time critical sensor data, can potentially provide more accurate data to functions requiring data generated on a time oriented basis than systems that operate on a minor cycle basis. However, using this method of timing does not preclude the use of minor cycles, which could be used to synchronize local processing element clocks and control the transmission of data on a periodic minor cycle basis.

### 3.1.4 Comparison of Information Transfer Systems

There are specific advantages and disadvantages to each of these ITS. The stationary master ITS has the greatest advantage of simplicity of operation. This type ITS has recently been used in aircraft systems (e.g., F-16, F-18, B-1, B-52 OAS). The SMITS has low processor overhead for synchronous messages and the control is centered at a single location, making integration testing the easiest of the three ITS. There are few disadvantages with this DAIS-like ITS, other than very poor asynchronous message handling capabilities. However, two other ITS offer better mission flexibility than this ITS. The nonstationary master is designed specifically for multimission applications, in which the secondary masters control the different aspects of a mission, such as specialized subsystems carried as pod or pallet mounted devices. The core avionics could remain standard while the mission-specific avionics under a secondary master control could be altered according to mission. The disadvantages of this system lie only in the overhead of transferring control from the primary controller through the secondary controllers and back to the primary control on a regular basis. If the number of controllers is limited to two or three, then this disadvantage is significantly ameliorated. There are several advantages provided by the contention system. The first of these is that the system is the easiest system to add multimission functions. Each new function could accept the core avionics data without impact to the system, since data are transmitted in a source oriented fashion. The data associated with mission specific functions would be generated by contending for the bus and transmitting the appropriate data to its own subsystems and to integration and display functions. The upgrading of an existing contention system, by adding new functions, can be done easily because of the asynchronous operation of the ITS. Several drawbacks to the contention system have been identified. These include the somewhat asynchronous nature of the system (even using minor cycles). Difficulties occur in the debugging of the system during integration because error conditions are not easily repeatable. This asynchronous system also could potentially require different feedback control algorithms for applications software than are normally used in totally synchronous systems. These algorithms do not allow the simplifying assumptions usually made in time dependent algorithms.

### 3.1.5 Task IV Conclusions

Advanced architectures using distributed microprocessors and multiple levels of buses are a reality and the AASMMMA program has provided a number of mechanisms to provide the required control. The DAIS executive can be rather easily modified to include multiple levels of bussing given the executive's modularity. The DAIS type of executive such as the SPSE could also be installed in the microprocessors of the 1980's, because both timing and space will allow for the inefficiencies inherent in a generalized executive such as DAIS. Avionic systems in the future will be constructed using many microprocessors and few miniprocessors, all interconnected via a number of buses. The entire system will be organized into a functional hierarchy of processing using one or more types of information transfer systems depending on the requirements of independence of function or multimission changes. If the multimission aspects of an avionic system are to be

implemented, then the three information transfer systems presented are excellent candidates. The stationary master is the best understood ITS and has multi-mission advantages if the applications software is designed for change. The nonstationary master is an excellent candidate for a pod-oriented multimission application. The contention access ITS is designed to be most flexible in terms of change, at the potential cost of higher initial integration checkout due to the asynchronous nature of the communication. Work on the contention system should continue in order to evolve and test it to the level of the master oriented protocols. A comparison of the three information transfer systems for both cost and dynamic performance remains to be done. Such an analysis would be of substantial benefit to answer remaining questions as to the actual merits of each of the functioning avionic information transfer systems.

### 3.2 TASK V - SPSE SUMMARY

The purpose of Task V was to design, develop, build and deliver a compact version of the DAIS executive which would function in a one processor system and support only synchronous bus communications. This executive, called the Single Processor Synchronous Executive (SPSE), has been tested and delivered to AFWAL.

The DAIS executive is a powerful, general-purpose executive for avionic systems which can support up to four processors communicating either synchronously or asynchronously over a MIL-STD-1553 bus. The DAIS executive was evaluated in a previous study (Evaluation of DAIS Technology Applied to the Integrated Navigation System of a Tactical Transport, AFAL-TR-1061) which indicated that the DAIS executive does an excellent job of servicing the requirements of a large and complex avionics system. For a small, less complex system, however, using the DAIS executive could be very inefficient. Substantial memory is required to provide features which may not be needed to support a small system. The large size of the full-capability DAIS executive has generated reluctance among some contractors to adopt it for use in new systems. Task V addressed this problem by analyzing the DAIS executive to determine which portions could be eliminated while still supporting the requirements of less complex systems. The results of this analysis were documented in the SPSE interim report.

The primary goals of Task V were to build a functional executive that:

1. Maintains the DAIS executive-to-applications interface.
2. Communicates on a MIL-STD-1553A bus.
3. Is coded in the JOVIAL high order language (J73/I).
4. Supports a synchronous bus communication requirement.
5. Supports the avionic system load for an AMST or modern tactical fighter aircraft.
6. Uses the DAIS support software such as LINKS, ALAP, PALEFAC and the PALEFAC preprocessor.

7. Requires substantially less memory than the baseline DAIS executive.

All goals were achieved.

All documentation of the SPSE was done using the existing DAIS documentation as a baseline. This included the System Control Procedures, the Part I (Design) specification, the Part II (Development) specification, the Test Plan, the Test Procedures, and the Test Reports. Since the basic structure of the DAIS executive was unchanged in the SPSE, the paragraph numbering of the DAIS Part I and Part II specifications was maintained in the SPSE documents to permit easy comparison of the two executives. Table 3-1 compares the main features of the DAIS executive and the SPSE.

The SPSE design, which provided the features shown in table 3-1, was reviewed and approved by AFWAL at the SPSE PDR and the SPSE CDR. Coding of the SPSE was begun following CDR.

The coding process was assisted by the use of two Boeing developed support software programs, SUBSORT and INGEUNS. SUBSORT accepts a list of all procedures called by a given procedure and generates a list of all procedures which call a given procedure. SUBSORT also performs a similar service with data items. INGEUNS creates a single compile-ready source file from a Universal Source File (USF). A USF contains the code for every version of the DAIS executive.

Once coding was completed, the SPSE modules were compiled and linked with the Model Avionics Program (MAP). MAP is a special applications program designed to place a controlled load on the SPSE, where the load is representative of a typical avionic system. The linked system was then installed in the DARTS laboratory (located in the Kent Space Center of the Boeing Company) for testing. Debug of the entire SPSE was completed in two weeks. Of this time, only about 3 days were spent in debugging the executable code itself, the rest of the time was spent in debugging the data base which had been modified for this system. Much of the success in being able to rapidly debug the SPSE software can be attributed to the use of the SUBSORT and INGEUNS support tools.

The SPSE was demonstrated at both Boeing and AFWAL, following the demonstration procedures developed as part of the contract. Both demonstrations were completely successful even though technical difficulties prevented either demonstration from being conducted in the exact manner specified in the demonstration procedures. One of the parts of the AFWAL demonstration was the test of the SPSE using the AFWAL Validation and Verification program. The SPSE passed the validation and verification test on the second attempt.

TABLE 3-1. COMPARISON OF DAIS EXECUTIVE AND SPSE

	DAIS (September 1979)	SPSE (March 1980)
Number of Mission Computers Supported	Multiple (to 4)	Single
Bus Communication	Synchronous and Asynchronous	Synchronous
Remote Terminals	30	30
Realtime Statements	Schedule Cancel Terminate Event Wait Time Wait Signal Read Write Broadcast Trigger Forced Read	Schedule Cancel Event Wait Signal Read Write
Executive Overhead for Synchronous System	X (where X represents the overhead for an arbitrary applications program)	0.60X to 0.95X
Memory Requirement	12,636 words	5,564 words

## 4.0 DETAILED PROGRAM FINDINGS

### 4.1 CANDIDATE INFORMATION TRANSFER SYSTEMS FOR MULTIMISSION APPLICATIONS

Three candidate ITS have been designed for future applications. They are: stationary, nonstationary and contention multiple access information transfer systems. These three ITS are described and compared below. For more detailed descriptions and comparisons see Appendices A, B, C and D of the first Interim Report Volume II, and the System Control Procedures and Part One Specifications written for each information transfer system. Appendix A, B, and C of this report details the rationale for the system control procedures written for each of the ITS.

#### 4.1.1 Stationary Master

The stationary master information transfer system (SMITS) is similar to DAIS in concept. The control site is centralized in one preselected terminal, designated the master bus controller, and this control point may be relocated to another terminal, designated the monitor bus controller, in the event of a master bus controller failure. Some of the advantages of the stationary master ITS are a simple/reliable BIU, hierarchical architecture capability, minimal risk of development, control dependability, effective bus capacity, low processor overhead for synchronous messages, use of the MIL-STD-1553B protocol and similarity to the DAIS ITS. The DAIS system specifications, and standards already developed were used as a baseline and were modified to provide the stationary master concept identified here. Some of the disadvantages of the stationary master ITS are multimission inflexibility, (because all messages are controlled from a single PE) and time critical responses. However, the stationary master represents a low risk to the overall development of an operating system for an aircraft.

##### 4.1.1.1 SMITS Control Procedures

The stationary master ITS control procedures (CDRL #18, Volume 1) were developed using the DAIS control procedures as a baseline. SMITS is significantly different from DAIS in four main areas:

- 1) DAIS has a single level bus network, while SMITS has a hierarchical (multi-level) bus network. The functioning of the two buses intersecting a single computer is assumed to be completely independent, which implies that there will be two independent BIUs reporting and operating with two different sets of minor cycles. The interfaces with two BIUs will involve two sets of interrupts to be serviced. The executive functions embedded in the PE expand to six combinations: master/master, master/remote, remote/remote, master/monitor, monitor/remote, and monitor/monitor. The SMITS studies, control procedures, and executive part 1 specification document the master/remote

combination. The master/master combination may be simplified to run with a single set of minor cycles for both sets of buses, although a single minor cycle is not a requirement.

Because the buses are to run independently, a number of assumptions can be made. The interprocessor service requests will be limited to a single bus level and any requests made between levels must be made through a service routine. Similarly message transmissions are limited to a single bus level. These hierarchical functions impact both the executive software (primarily master executive) and the associated executive support software (viz. PALEFAC).

- 2) The DAIS program uses federated minicomputers, while SMITS uses distributed microcomputers and minicomputers. Distributed processors allow some of the control to be removed from the master executive to the remote processors. Future aircraft systems will buffer serial digital interfaces at remote terminal interfaces by using microprocessors and will not control the operation of these devices via the bus as was done in DAIS). Similarly, mass memory control was removed to a remote terminal, so that the master executive duties are reduced to requesting particular data and waiting until the data is received. The microprocessor controlling the disk is responsible for the error responses with respect to any disk control problems, thereby alleviating master executive complexity (and centralization of responsibility).
- 3) DAIS employs its own version of MIL-STD-1553A protocol, while SMITS employs MIL-STD-1553B protocol. The change in protocol to the current military standard provided some (but certainly not all) motivation to change the disk and serial digital channel operations from the DAIS mechanism. The DAIS method was to provide additional communication via the 1553A status word, which is forbidden in the 1553B status word. Also affected are asynchronous messages in general. The SMITS offers asynchronous communications both with and without a concluding handshake message, and provides the mechanisms by which the communications can be implemented with each option. SMITS also allows for the broadcast of messages, but does not use the dynamic bus control mode codes used in the SMITS.
- 4) DAIS uses the IBM-DAIS BCIU, while SMITS uses the AFWAL/ADH sponsored BIU chipset. The two chipset will allow microprocessors (again a small set of chips) to be interfaced to a 1553B bus with a minimum of expense. One chip will allow a processor to interface only as a microprocessor, and the two chips together allows bus control to transpire. This chipset also has a number of spare bits in its command words so that if it were to be expanded in capability, it could satisfy another aspect of the multi-mission role being investigated by the AASMMA study: that of life cycle evolution. For each different 1553 protocol (e.g. 1553B, 1553A/DAIS, 1553A/B52, 1553A/F-16), if a bit in the bus channel control word were to specify the variant of 1553 protocol, and interrupt the PE only on a status word indicating an error or desired service, then equipment from all of the various systems using 1553 could be intermixed using these chips to the master executive to the mixed equipment bus. Any additional equipment could be added as either 1553B compatible (as the preferred protocol) or any other 1553 compatible equipment.

Appendix A presents a paragraph by paragraph documentation of rationale for changes to the DAIS control procedures and for the new and redesigned procedures which make up the SMITS control procedures.

#### 4.1.1.2 SMITS Executive Functional Description

This section describes the requirements of the SMITS executive as defined in Volume 1 of CDRL #17. The SMITS executive software is divided into two major functions: the local executive and the master executive. In general, the local executive controls processes involved with a single processor, while the master executive controls processes concerned with the interaction of many processors. Each bus or bus pair in the hierarchy has a master executive to control the functioning of the system communication at that level.

##### Local Executive

Each of the processors contains a local executive very similar to the DAIS executive. This local executive controls the state of the realtime entities existing within its processor, specifically tasks, global copies, and copies of events (which, like COMPOOL blocks, may exist in multiple copies, one in each processor within which the event is referenced).

The local executive performs services requested by tasks in realtime statements. Since a realtime statement executed in one processor may affect the state of a realtime entity in another processor, the local executive must be able to send asynchronous messages requesting services in other processors (e.g., to SCHEDULE a task or update a compool block).

In addition, the local executive must receive such requests from other processors, and service them properly.

Unlike asynchronous processes, synchronous processes are basically under the control of the master executive, since synchronization is a process involving all of the processors on its bus. However, the local executive must also participate in synchronous processes, by signaling minor cycle events and preparing for the reception and transmission of synchronous COMPOOL block update messages.

Finally, the local executive must be capable of starting its processor, and of recognizing and processing errors that may arise.

##### Master Executive

The master executive controls communication between the separate processors and remote terminals on its bus. This communication exists only in the form of messages which can be sent across the bus. Thus, one major function of the master executive is bus control.



Each remote terminal or processor can request to send asynchronous messages. There are also synchronous messages which must always be sent at a given period and phase of a minor cycle. Critically timed messages may be sent at a specified mission time. In addition, the master executive has its own messages which are used to determine the correct functioning of the processors and RTs on its bus.

The primary function of the master executive is to control the transmission of these messages. The secondary function is to take corrective action when one of these messages appears to have been sent incorrectly. The corrective action taken may be to resend the message under carefully controlled conditions. If this fails, the error cause is assumed to be hardware which has ceased to function properly. Either functionally redundant hardware must be invoked or the scope of the mission must be changed. This is known as system configuration management. System configuration management keeps track of the status of all processor and bus-related hardware to be used during this mission, and determines the operations to perform when a hardware element fails. It may also have to report these failures and reconfigurations to other buses in the hierarchy.

The master executive as discussed above is a set of functions which exist in one processor on each bus and possibly within a backup processor. This processor is called the master processor and the BIU attached to that processor is called the master BIU.

To allow for the possibility that the master processor or the master BIU may fail, a second master executive can exist in another processor. This second master executive is called the monitor executive. The master executive must periodically send a message to the monitor which informs the monitor that the master executive is still functioning. If the monitor does not receive this message, it switches to master executive mode and takes control of the bus. (The switchover may also be accomplished using discretes between the two bus controllers). If the mission is in a noncritical phase, the monitor will automatically reconfigure around the failed processor via the warm start phase. The monitor will take control of the bus and initiate a set of applications tasks and bus commands to ensure that the bus elements perform the optimum processing possible under the circumstances. This mode of processing is called degraded mode. Degraded mode is also entered should the master be unable to reconfigure around a remote processor/BIU failure during a critical mission phase.

The purpose of backup mode is to prevent any processing delay caused by a failure during a critical mission phase. During backup, the monitor processor causes only certain specified critical functions to be performed. The pilot always has the option to restart or reload the system by using the processor control panel.

There is a third part of the master executive which is the system loader. The function of the system loader is to load the mission software from system mass memory into all the processors within the currently specified configuration. The system loader receives control of the bus from the ROM which was the bus controller after processor power up.

The ROM loader in each processor gains control of that processor upon power up. After completing self tests and processor verification, the ROM loader examines the bus and if it finds no activity it will assume control of the system, thus

becoming the controller. It then polls all processors in the initial configuration to determine if one of the processors has a valid system loader. The ROM loader will then supervise the loading of its processor with the system loader from mass memory if a load is necessary. Upon completion of the load or notification of a valid load, the ROM then hands control of the system to the system loader.

If when the ROM loader comes up, the bus is inactive, it will then go into an idle loop until one of the following happens:

1. Commands are received from another ROM loader.
2. Commands are received from the system loader.
3. A predetermined length of time has passed, in which case the ROM loader will again attempt to become the controller.

#### 4.1.2 Nonstationary Master

The nonstationary master information transfer system (NSMITS) is based upon modified SMITS system control procedures. These procedures have been modified to support a nonstationary master architecture which uses a round robin scheme to determine the processor that will next control the bus. The NSMITS, like the SMITS, is an extension of DAIS to allow for MIL-STD-1553B and a hierarchical bus structure. The result is a system which closely resembles DAIS in its philosophy and application, but provides additional capabilities for the multimission applications envisioned for advanced digital systems in future aircraft.

The nonstationary master information transfer system uses a round robin bus control transfer scheme to provide the system designer increased flexibility in developing a system which can support any one of several different missions. In a nonstationary master system, an aircraft can be readied for a new mission (e.g., a reconnaissance mission after an attack mission) without modifying any of the bus control tables or procedures. The only overt action needed to implement the change is to reload the multimission processor with software appropriate for the new mission and interconnect the new avionics subsystems. This change should be completely transparent to the rest of the system. Therefore, all mission dependent bus messages are initiated by the multimission processor when it gains control of the bus. The multimission processor uses standard bus commands to collect the necessary data from the other devices on the bus. No changes are needed in any of the instruction lists or bus control tables resident in the other processors. Part of the nonstationary philosophy includes a core of processors which perform the general processing functions and multimission processors which will be modified as required for new missions. The advantage of this philosophy is that the core of processors are isolated from the multimission processors, which minimizes change from mission to mission.

This ITS documented in Task IV differs from the nonstationary master ITS defined in Task I in the way bus control transfer is performed. The Task I bus control was passed via a polling mechanism. Instead of polling each potential master to find the one with the highest message priority, bus control transfer is now performed in a round robin sequence in which each master transfers control to a

predefined new master. Each new master operates as if it were a stationary master until its transmission interval is complete. The NSMITS control procedures provide for hierarchical bus structures and MIL-STD-1553B. The reason that the change was made from polling to round robin control was twofold: (1) the overhead and hardware required to implement a polling scheme is very high, and (2) the number of bus controllers for feasible multi-mission applications are two or three in number. Round robin control on such a small number of PEs is much simpler than polling.

#### 4.1.2.1 NSMITS Control Procedures

The nonstationary master information transfer system control procedures (CDRL #18, Volume 2) are based on the SMITS control procedures but provide additional capabilities for multimission applications. Bus control transfer between master processors on the NSMITS bus occurs in a round robin sequence. One processor, designated the primary master, performs minor cycle synchronization for the NSMITS and is responsible for configuration and bus control management for non-multi-mission-specific equipment. The other master processors on the NSMITS bus are referred to as secondary masters. One of these secondary master processors can be designated as monitor and will contain a copy of the executive and critical application software contained in the primary master.

Minor cycle synchronization of the NSMITS bus is performed by the primary master while it is master mode and is received by the secondary masters while they are in remote mode. After synchronizing the bus, the primary master begins bus activity for the minor cycle by executing its synchronous instruction list (SIL). After the end of the SIL, the low priority asynchronous bus instruction list, and the status polling list, bus control is transferred to the next processor, and the primary master converts to a remote. The next processor then executes its bus instruction lists for this minor cycle. This process is continued until each processor in the NSMITS has become a bus controller and performed its required message processing. The last processor in the round robin chain returns bus control to the primary master, which waits until the end of the current minor cycle before starting the new minor cycle. The bus control transfer protocol uses the dynamic bus control mode code of MIL-STD-1553B. If timing and control authority is critical with respect to the primary master, then a discrete could be extended between the primary master and the other master(s). The discrete would indicate a subordinate master's BIU must cease transmission because of the beginning of the next minor cycle.

The primary master also has the function of bus control management. This function is to monitor the round robin sequence of bus control transfer and determine the processor at fault if a bus control transfer fails.

NSMITS supports an optional mass memory subsystem to support the capabilities to load/reload program modules and mission dependent data and to record data for post flight analysis. The mass memory for the nonstationary master system is connected only to the NSMITS bus instead of being an interbus processor as in the SMITS. This architectural change enhances the multimission philosophy of NSMITS.

NSMITS configuration identification and software module loading occurs in the lowest bus levels first and proceeds up the hierarchy until the global bus is identified and loaded. This allows a higher bus level to change its configuration based on the configuration of lower levels.

#### 4.1.2.2 NSMITS Executive Design

This section describes the requirements of the NSMITS executive as defined in Volume 2 of CDRL #17. The NSMITS executive software is divided into three major functions: the local executive, the master executive and the system control executive. In general, the local executive controls processes involved with a single processor, while the master executive controls processes concerned with the interaction of many processors. The system control executive is involved in system-wide management on the NSMITS bus, and has the error management and the minor cycle control which the stationary master of the SMITS. The local executive is essentially identical to the SMITS local executive. The master executive has the same functions as in the SMITS, with the exception that bus control error handling is included and minor cycle control is executed. The system control executive software is responsible for overall bus management and for minor cycle synchronization. Although the SMITS control procedures and executive part one specification could be described in terms of changes from the SMITS the documents are fully self-contained. The appendix B discusses some of the specific changes and the rationale for those changes from SMITS.

##### System Control Executive

The system control executive will reside only in the primary master processor and the monitor. Its main function is to detect the failure of bus control transfer between master processors on the NSMITS bus. Each master processor transmits a message to the primary as the first step to transfer bus control which updates a bus control transfer table. When a processor fails to accept bus control, the failure is updated to configuration management.

To allow for the possibility that the primary master processor or its BIU may fail, a second master executive and system control executive can exist in another processor. This second primary master processor is called the monitor. The primary master processor must periodically send a message to the monitor which informs it that the primary master processor is still functioning. If the monitor does not receive this message, it must switch to master executive mode and take control of the bus and assume the responsibility of the primary master. If the mission is in a noncritical phase, the monitor will automatically reconfigure around the failed processor via the warm start procedure. However, if the primary master should fail during a critical mission phase, the monitor will take control of the bus and initiate a set of applications tasks and bus commands to ensure that the bus elements perform the optimum processing possible under the circumstances. This mode of processing is called degraded mode. Degraded mode is also entered should the master be unable to reconfigure around a remote processor/BIU failure during a critical mission phase.

The purpose of backup mode is to prevent any processing delay caused by a failure during a critical mission phase. During backup, the monitor processor causes only certain specified critical functions to be performed. The pilot always has the option to restart or reload the system by using the processor control panel. This is called manual reconfiguration.

#### 4.1.3 Contention Multiple Access Information Transfer System

In the contention multiple access information transfer system (CMAITS), there are no unique bus masters, rather each device contends for use of the transmission media. When a device wishes to transmit, it will sense the communication activity on the transmission media to determine if a transmission is allowed. If activity is detected, the device will wait until the transmission ceases, then the device will begin a random delay associated with its queued message's priority before attempting to transmit. If the transmission media has remained free of communication activity for the duration of the delay, the device initiates transmission. Using this approach, multiple devices can attempt communications on the transmission media simultaneously. If this occurs, a collision is said to have occurred, and the transmissions are terminated and rescheduled for transmission later. Once a device acquires the transmission media and begins communication which does not result in a collision, it may transmit until a given length of time expires. During this period of time, called the transmission interval, as many complete contiguous messages as can be transmitted in the time interval will be scheduled and transmitted.

The contention mechanism discussed here allows considerable flexibility in system integration. This ITS has the advantage of easily providing for new, modified or multimission oriented sensors. Because of the independent nature of the processors, modifications to the system will primarily affect the processors responsible for control of the modified section.

There are several advantages provided by the contention system. The first of these is that the system is the easiest system to add multimission functions. Each new function could accept the core avionics data without impact to the system, since data is broadcast in a source oriented fashion. The data associated with mission specific functions would be generated by contending for the bus and transmitting the appropriate data to its own subsystems and to integration and display functions. The upgrading of an existing contention system, by adding new functions, can be done easily because of the asynchronous operation of the ITS. Functions by necessity and design are as loosely coupled as possible. This loose coupling will allow for easier upgrade of capability than systems that are tightly coupled by the information transfer system using synchronously generated messages within a given period of time. This contention organization also is most closely aligned with the method of integration used today by an avionics integrator who purchases subsystems and integrates them using the ITS integration process. Because the contention algorithms are based on message priority, the highest priority messages can be transmitted within a shorter time than in other types of multiple control mechanisms. This capability can be very important in advanced vehicles where rapid responses are required.

Several drawbacks to the contention system have been identified. These include the somewhat asynchronous nature of the system (even using minor cycles). Difficulties occur in the debugging of the system during integration because error conditions are not easily repeatable. This asynchronous system also could potentially require different control algorithms than are normally used in totally synchronous systems. These algorithms do not allow the simplifying assumptions usually made in time dependent algorithms. The advantage of time tagged data is that it does provide the capability to modify data which is collected synchronously and transmitted asynchronously. The contention system bus allocation algorithms involve the detection of an inactive bus, a random waiting period and the transmitting of a message sequence on the bus. This procedure requires less bus overhead than a stationary master system.

#### 4.1.3.1 CMAITS Control Procedures

This section describes some of the differences between the DAIS baseline and the contention multiple access information transfer system (CMAITS). The control procedure document (CDRL #18, Volume 3) is entirely different from the DAIS control procedure for the simple reason that the contention information transfer control hardware and software are very much different from those used in the DAIS program.

The contention system was designed around the concept that the real solution to multi-mission applications is independence of function. The most independent set of devices operating in an integrated multiplexing system is a set of devices which contend for control of the bus when the device has information to transmit. Once control is obtained, data is broadcast to all devices which have that message identifier in its receive address list. The sequence by which the various devices assume bus control is random, making each device potentially independent from all other devices. From these simple assumptions an information transfer system has been designed. The operation of this ITS will be discussed below briefly in the three areas of contention access, broadcast control, and system control, and is discussed in more detail in Appendix C of this report. The full discussion can be found in the Contention, Multiple Access Information Transfer System Control Procedure document (CDRL #18, Volume 3).

#### Contention Access

The contention access portion of the ITS is totally different from DAIS. Whereas DAIS has a single controller which has absolute authority over the bus, each CMAITS element may capture control of the bus and hold it for a period of time. The BIU hardware to perform the control is approximately twice as complex as the DAIS or SMITS BIU's. The BIU's normal functions must include all those in the DAIS bus controller plus the following: (1) listen for a quiet bus, (2) determine whether a message is available to transfer, (3) determine the highest priority message to transfer, (4) wait until a (random) time has elapsed, based upon the priority of the message, (5) commence transmission while concurrently receiving in order to detect whether a transmission is occurring simultaneously, (6) cease transmission and wait if a simultaneous transmission occurred. The BIU

also has additional functions for reception of messages. The BIU must compare the address on the incoming message with a list of message addresses to determine whether the message should be accepted by it. If a message is also determined to be invalid because of bad parity, etc, the BIU may respond with an invalid message response at the conclusion of the message transmission sequence.

#### Broadcast Control

The broadcast function is used in the SMITS as an extension of the DAIS bus control function. It is further extended in CMAITS to include message identification rather than source and destination definition. The major impact is on the BIU hardware which must be able to discriminate which of the messages are destined for its PE and then to map the message appropriately into the PE's memory.

#### System Control

The control of the CMAITS is not centered within a single controller, although that option is not excluded. Transmission control is totally distributed because of the bus access mechanism. The task control, error control and configuration control are not totally distributed. Because of the synchronous nature of some of the avionics computations, two or more of the PE's in a CMAITS may wish to proceed in lockstep via minor cycle synchronization. Therefore, CMAITS can consist of a number of different PE's each controlling its set of PE's and their tasks at different minor cycle rates. Each of these computing groups is called a sphere of control. The same PE performing the synchronization function is also responsible for the configuration management and error handling of equipment within the sphere. In many ways, each sphere has the same configuration management characteristics as DAIS. However, if a PE fails, the redistribution of all of the tasks can be accomplished more readily because of the broadcasting of the messages and content-identification of the messages. Therefore the destination of messages do not have to be redefined, but rather only the relocation of critical tasks.

#### 4.1.3.2 CMAITS Executive

This section describes some of the differences between the DAIS executive and the requirements for the CMAITS executive as defined in Volume 3 of CDRL #17. The differences are almost totally in the master executive (bus control) portion of the executive. The local executive, with its task control functions, remains very much the same as in the DAIS executive.

The CMAITS executive is a synchronous executive which operates either upon minor cycle events from a sphere controller or controls its own minor cycles via its clock. The executive to applications communication interface is the same as in DAIS. CMAITS may also have distributed tasks which require that the executive communicate an event set in one PE to the remainder of the PE's, as is done in DAIS by a set of command/response messages. While this is allowed, a

significantly smaller sized executive can be used for a singular processor which operates synchronously using its own clock. The master executive BIU controller is substantially different from DAIS. The BIU interface is different because of the additional capabilities required in the BIU. Several bus priority communication lists exist and must be scanned by the BIU between message sequence transfers. From highest to lowest, the priority of transmission is (1) data messages in response to request messages, (2) critically timed (trigger) messages, (3) synchronous messages, and (4) asynchronous messages. The messages in each of these four communication lists must be managed by the CMAITS master executive, while the DAIS executive has only one list which must accommodate all four types of messages. The executive must also initialize the BIU RAM with all desired message identifiers and specify their indexes into the message pointer list so that each message can be deposited via DMA. Another bit in the RAM will specify whether the PE should be interrupted upon the arrival of the message. If an interrupt is caused, an event is created which is passed to the local executive for action by any tasks waiting on that event.

The error handling of the master executive at the most critical level involves the cessation of all bus transmissions and interrogation of devices suspected of failure. Redistribution of functions according to the reconfiguration procedures are also accomplished by the master executive of both the sphere controller and the individual PEs.

## 4.2 ANALYSIS OF INFORMATION TRANSFER SYSTEM PERFORMANCE

### 4.2.1 Message Wait Time

Message wait time or latency is the amount of elapsed time from when a message becomes ready to transmit in one device to the completion of receipt by a second device. The analysis will consider a message as being either a time-critical message (i.e., trigger message) or nontime-critical. In any given system the trigger message is assumed to be an important component, but will comprise a small percentage of the total message traffic. A weapon delivery message is an example of a trigger message. Figures 4-1 through 4-4 show the effect of several variables on message latency for the three ITS.

#### 4.2.1.1 Stationary Master ITS (SMITS)

When using SMITS the latency of a trigger message is determined by the length of time it takes for the bus controller to give a particular device the right to transmit the message request plus the time it takes for the master to handle the trigger message request and allow the device to transmit the trigger message. The sequence of events associated with the processing of a trigger message are:

1. Recognition of the service request bit in the status word of MIL-STD-1553B
2. Transmission of mode code to requesting device (transmit vector word)



3. Receipt of service vector information
4. Transmission of command requesting device to transmit the trigger message
5. Transmission of the trigger message
6. Reset to normal traffic which has been interrupted.

All six of these activities could (and should) be performed by the bus interface unit (BIU), however current implementations of BIU hardware lack the required sophistication. The CPU must be interrupted to handle the service requests generated in steps 1 and 3 above. The service of a trigger message will require:

- 2 interrupt services (maybe 1 more for link back to schedule message list)
- 6 message gaps/response times
- 7 command/status words/vector
- L data words where L is the length of the trigger message

Each interrupt will require the machine state to be saved, the service routine address to be determined, the service routine executed, and the machine state to be restored.

A fixed time of 50 microseconds will be used to estimate the processing of each interrupt. The length of a trigger message is assumed to be 10 words. The message gap is dependent on bus bandwidth. At low data rates on an optical bus, a message gap of 4 bit times can be attained. At higher bandwidths, the propagation delay ( $\frac{1}{2}$  us per 300') becomes significant and message gaps must be increased accordingly. The command, data and status words each require 20 bit times per word. The total message service time is then:

$$\text{Message Service Time} = 2(50) + 6\left(\frac{1}{2} + \frac{4}{B}\right) + (7 + 10)\left(\frac{20}{B}\right) = 103 \text{ us} + \frac{364}{B}$$

(Equation 1)

where:

B is the bus bandwidth in MHz. In this report, the term bus bandwidth means the maximum bit rate on the bus.

The other component of trigger message latency is the delay associated with obtaining the right to transmit the trigger message. This delay is dependent on the system configuration and organization. A minimum delay configuration would allow the polling after every message transmission of all devices that have the potential for a trigger message. A more reasonable configuration would be to poll one potential device after each message transmission or to poll each device one or more times during a minor cycle. For the analysis, the assumption of one device being polled after each message transmission will be used to illustrate the minimum potential latency of a SMITS system. If polling is performed on a minor cycle basis, say once every 7.8 or 15.6 ms, then the latency would be an order of magnitude greater than the system which is polled after every message. On the average, the device with the trigger message will be the middle one

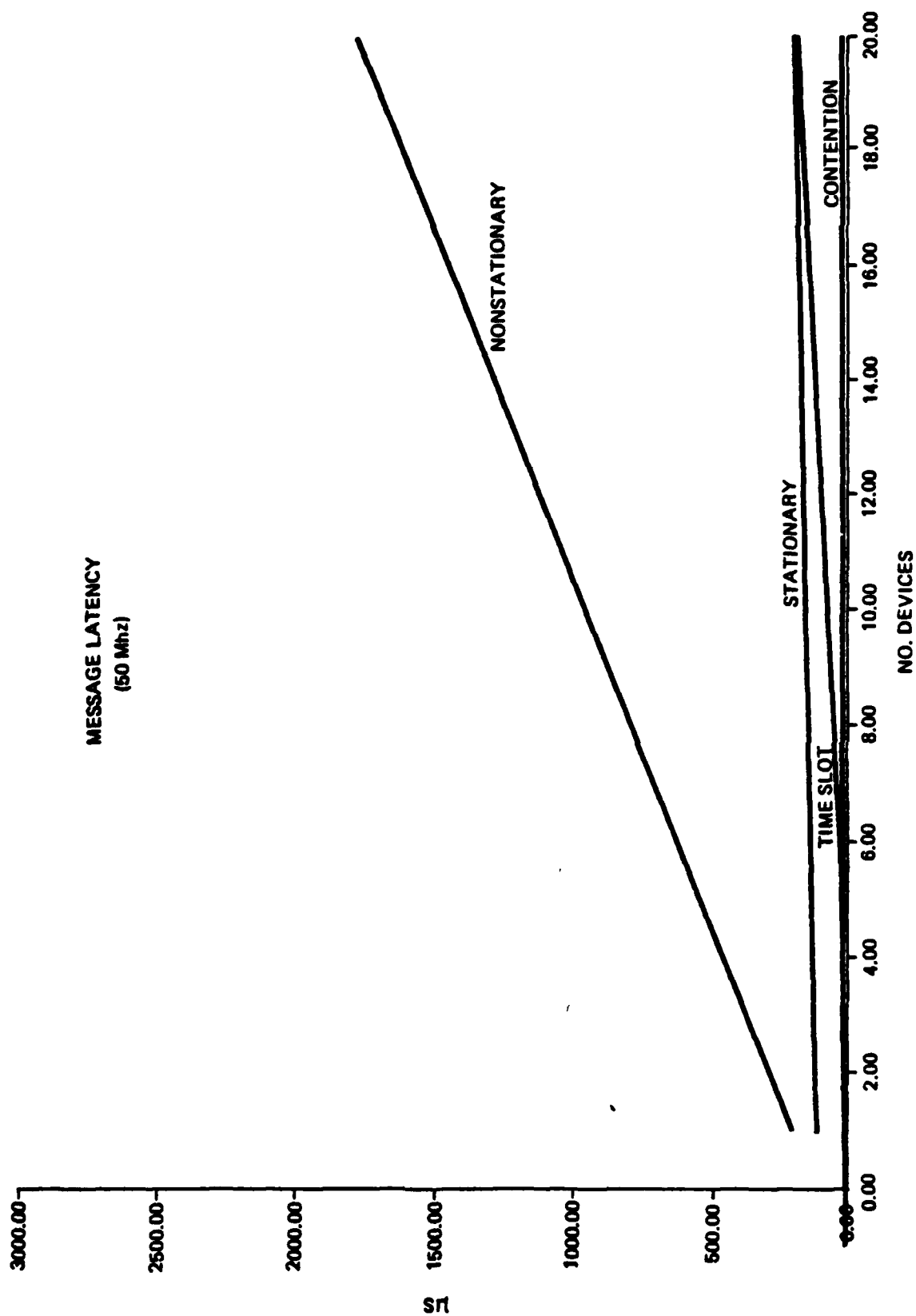


Figure 4-1. Effect of Number of Devices on Message Latency

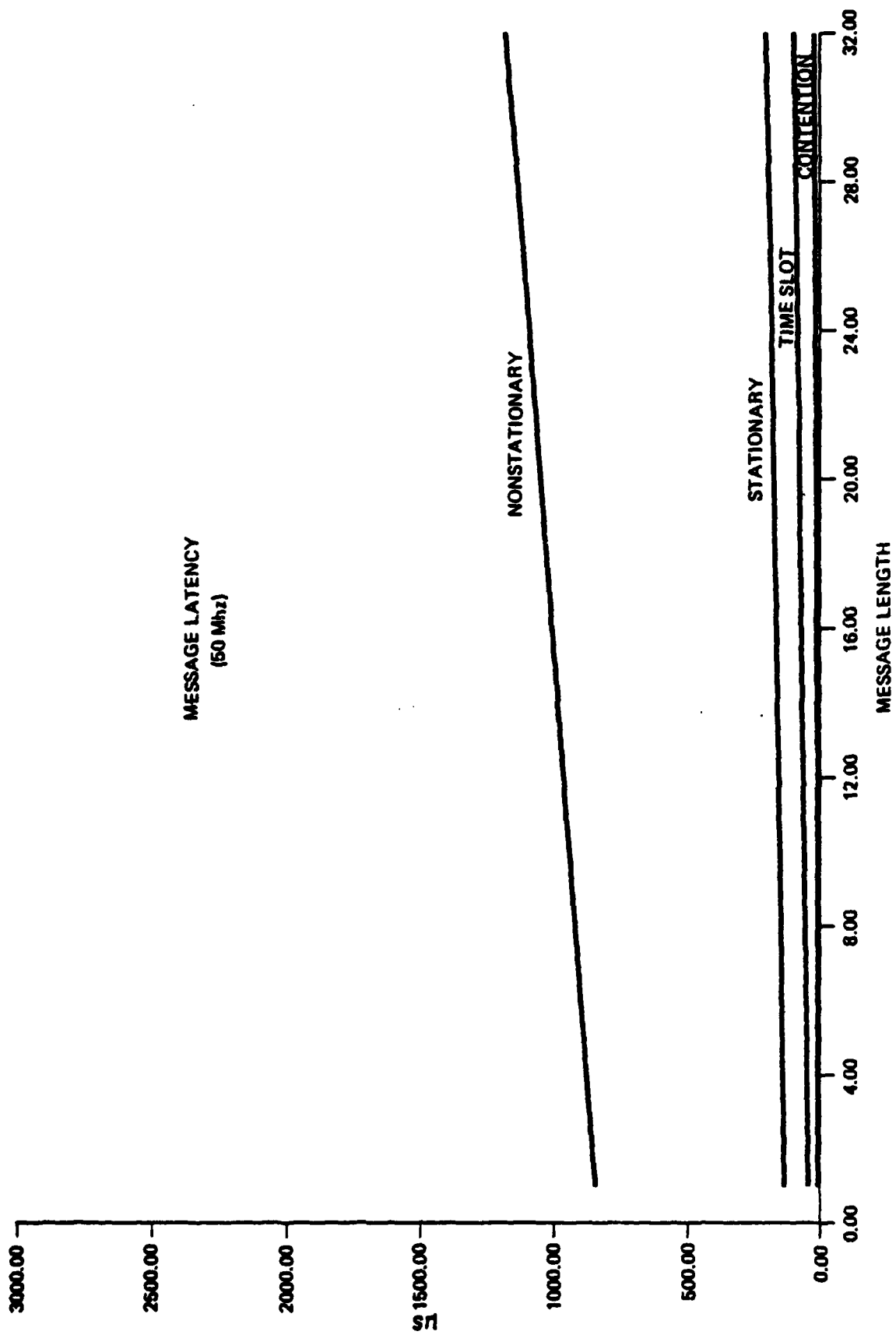


Figure 4-2. Effect of Message Length on Message Latency

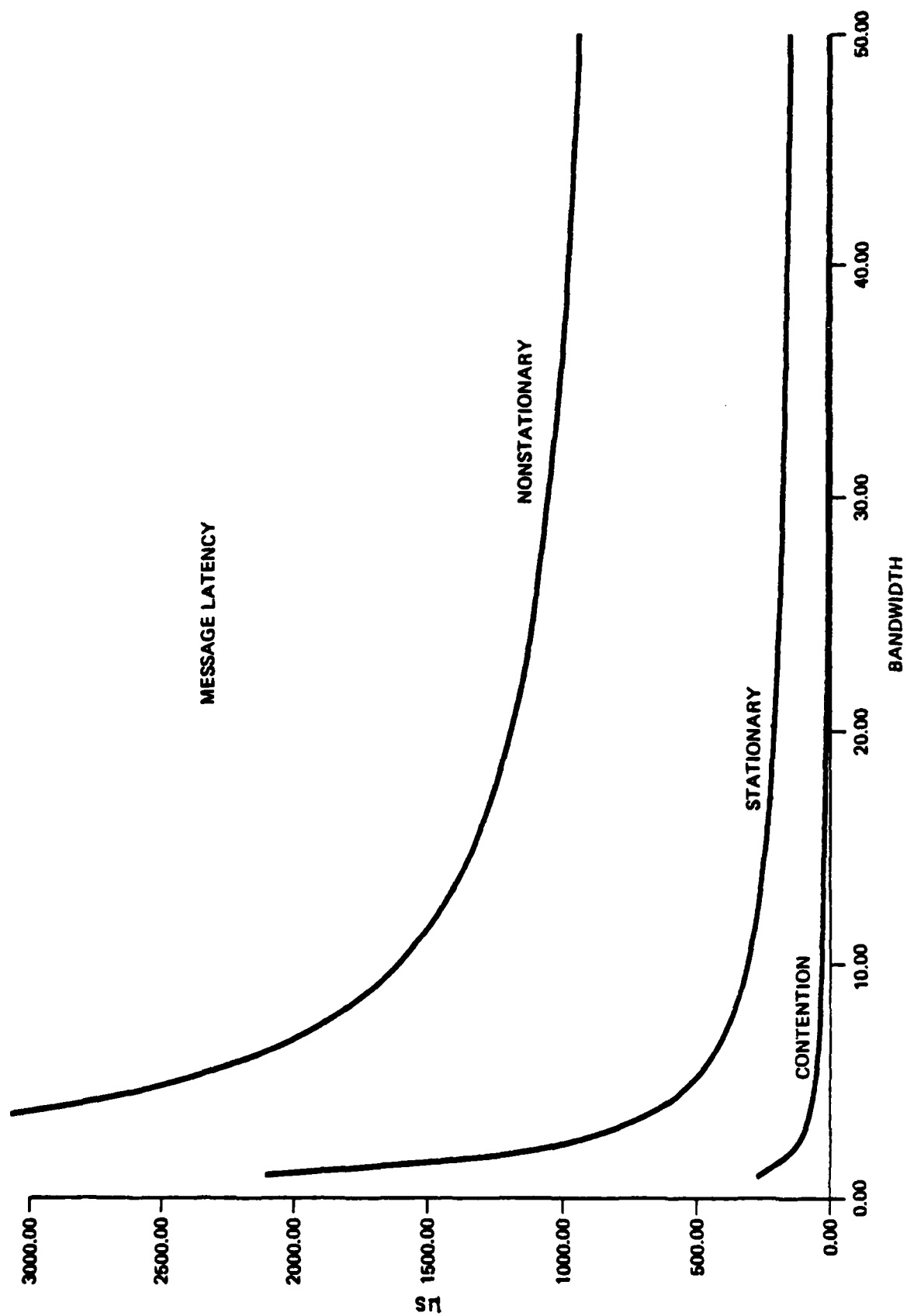


Figure 4-3. Effect of Bandwidth on Message Latency

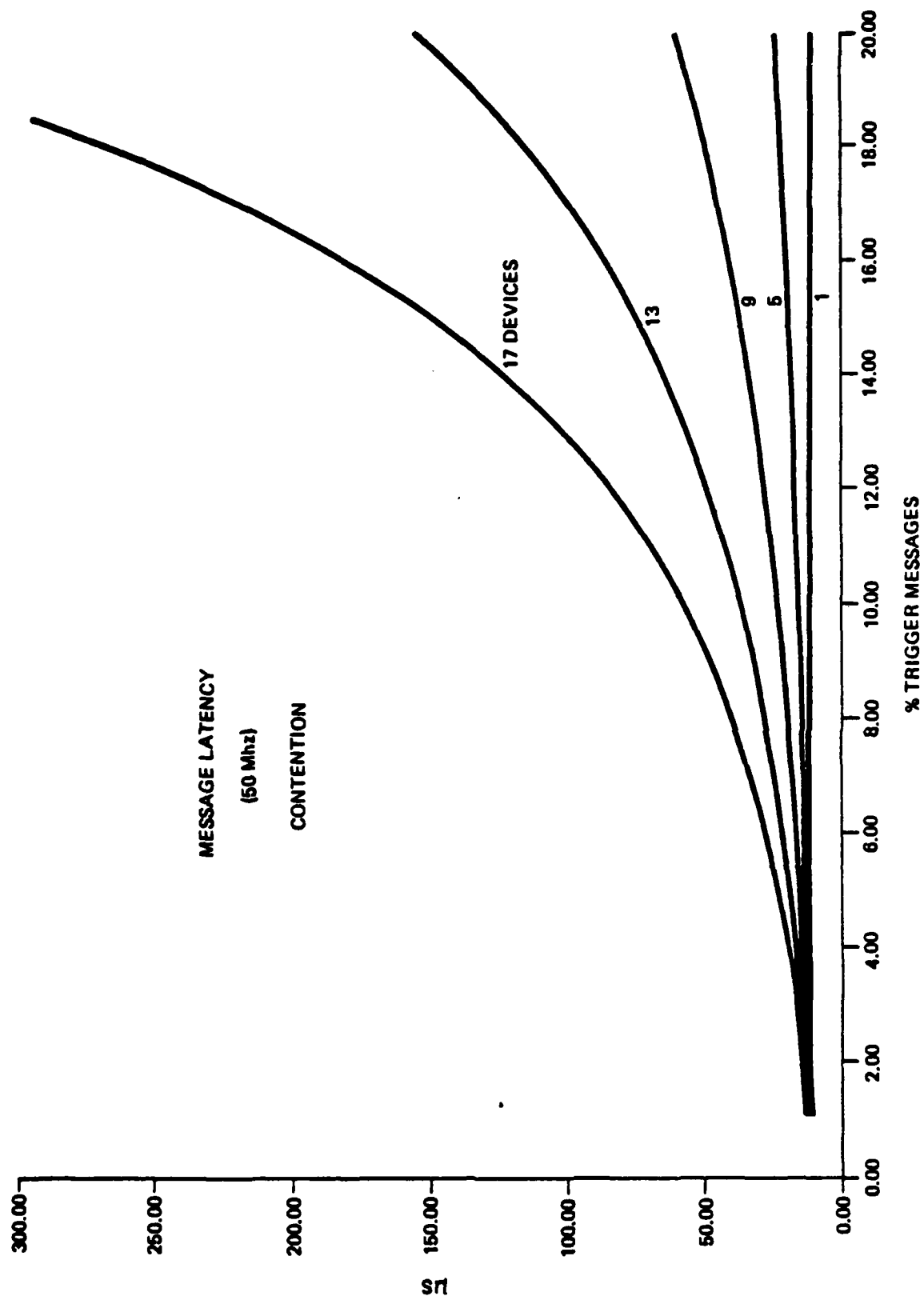


Figure 4-4. Effect of Trigger Messages on Contention Message Latency

polled. The required time in this case is  $(n/2 \text{ message times}) + (n/2 \text{ polling times})$ , where  $n$  is the number of devices with potential trigger messages. Each polling is made up of a command word and a status response which requires two message gaps and two word times. A message time is dependent on the type of message that is being sent in the system. The message mix that is assumed for this analysis is shown below:

1. Master - Remote or Remote - Master	50%
2. Remote - Remote	30%
3. Master - Remote (Broadcast)	10%
4. Remote - Remote (Broadcast)	5%
5. Mode Code Without Data	2%
6. Mode Code With Data	1%
7. Mode Code Without Data (Broadcast)	1%
8. Mode Code With Data (Broadcast)	1%
	<u>100%</u>

The total time to acquire the right to transmit the trigger message is then:

$$\frac{(Q+1)}{2} \sum_{i=1}^8 M_i \left[ G_i \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{20}{B} (S_i + L) \right] + 2 \left( \frac{1}{2} + \frac{4}{B} \right) + 2 \left( \frac{20}{B} \right)$$

where:

- M is the mix percentage
- G is the number of message gaps
- S is the number of command and status
- L is the average message length
- B is the bus bandwidth
- i is the mix index
- Q is the number of potential trigger message devices

The total message latency is the sum of service time and recognition time which is:

SMITS Latency (us) =

$$\frac{(Q+1)}{2} \sum_{i=1}^8 M_i \left[ G_i \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{20}{B} (S_i + L) \right] + \frac{(Q+1)}{2} \left( 1 + \frac{44}{B} \right) + 103 \left( \frac{364}{B} \right)$$

#### 4.2.1.2 Nonstationary Master ITS (NSMITS)

Message latency for NSMITS is determined by the same message delays as SMITS plus the delay associated with the transfer of bus control. Because each potential bus controller does not have knowledge of every device capable of sending a trigger message, bus control transfer must occur prior to service of some trigger messages. The round robin protocol of NSMITS gives the lowest possible trigger message service time.

$$\text{NSMITS Latency} = \frac{5(Q+1)}{2} \sum_{i=1}^8 M_i \left[ G_i \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{20}{B} (S_i + L) \right] + \frac{(Q+1)}{2}$$

$$125 + 4 \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{5 + (20)}{B} + Q \left( 1 + \frac{44}{B} \right) + 103 + \frac{364}{B} =$$

$$\frac{5}{2} (Q+1) \sum_{i=1}^8 M_i \left[ G_i \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{20}{B} (S_i + L) \right] + \frac{102Q + 422}{B} + \frac{129Q}{2} 104$$

#### 4.2.1.3 Contention Multiple Access ITS (CMAITS)

The CMAITS protocol distinguishes trigger (highest priority) messages from other messages by allowing only trigger messages to be transmitted in the first time slot available for message transmission after detecting the bus's availability. This assumes that there are no command-response messages required in the system. If a trigger message is ready to be transmitted, it will have the first opportunity to use the bus. Since there are so few trigger messages, the chance of a trigger message collision is very small. With this scheme the latency of a trigger message is the time required to detect an inactive bus plus the message transmission time plus any delays due to collisions. The availability of the bus can be determined within two propagation delays or 1 us. Message transmission time was defined earlier in equation 1. Collision delays are a function of the number of devices capable of transmitting trigger messages and the probability of a trigger message being transmitted. The expected collision delay is:

$$E(C_D) = 1 \left[ 1 - \frac{1}{2} \left( 1 - \frac{1}{2}P \right)^{Q-1} \right] + 4 \left[ 1 - \frac{1}{2} \left( 1 - \frac{1}{2}P \right)^{Q-1} \right]^2 +$$

$$8 \left[ 1 - \frac{1}{2} \left( 1 - \frac{1}{2}P \right)^{Q-1} \right]^3 \dots = \frac{\left[ 1 - \frac{1}{2} \left( 1 - \frac{1}{2}P \right)^{Q-1} \right] \left[ 2 - \frac{1}{2} \left( 1 - \frac{1}{2}P \right)^{Q-1} \right]^2}{\left[ \frac{1}{2} \left( 1 - \frac{1}{2}P \right)^{Q-1} \right]^2}$$

where:

P is the proportion of messages that are trigger messages

Total latency then is:

CMAITS Latency =

$$1 \text{ us} + \sum_{i=1}^8 M_i \left[ G_i \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{20}{B} (S_i + L) \right] +$$

$$\frac{\left[ 1 - \frac{1}{2} \left( 1 - \frac{1}{2}P \right)^{Q-1} \right] \left[ 2 - \frac{1}{2} \left( 1 - \frac{1}{2}P \right)^{Q-1} \right]^2}{\left[ \frac{1}{2} \left( 1 - \frac{1}{2}P \right)^{Q-1} \right]^2}$$

#### 4.2.2 Information Throughput

The number of 16 bit data words that can be received per microsecond by devices in the ITS defines the throughput of the ITS. Word overhead, message overhead and the loss due to collisions reduce the effective throughput of all ITS. Figure 4-5 shows the effect of bandwidth on throughput for the three ITS.

##### 4.2.2.1 Stationary Master ITS (SMITS)

The throughput of the SMITS is computed as the number of messages that can be sent per microsecond and the percent of data contained in each message. A message time, as defined in section 4.2.1.1 is:

$$\text{Message Time} = \sum_{i=1}^8 M_i \left[ G_i \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{20}{B} (S_i + L) \right]$$

If the average message contains L data words and each 20 bit word contains 16 information bits then the throughput of SMITS is:

$$\text{SMITS Throughput} = \frac{\frac{16}{20} L \text{ words}}{\sum_{i=1}^8 M_i \left[ G_i \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{20}{B} (S_i + L) \right]}$$

##### 4.2.2.2 Nonstationary Master ITS (NSMITS)

The same factors that affect throughput of the SMITS also affect throughput of the NSMITS. However additional overhead is required to accomplish the transfer of bus control. This transfer requires 5 words and 4 message gaps to satisfy the transfer protocol and 125 microseconds for a new processor to gain control. This control transfer will be assumed to be performed after every fifth message transmission. The throughput of the round robin NSMITS is then:



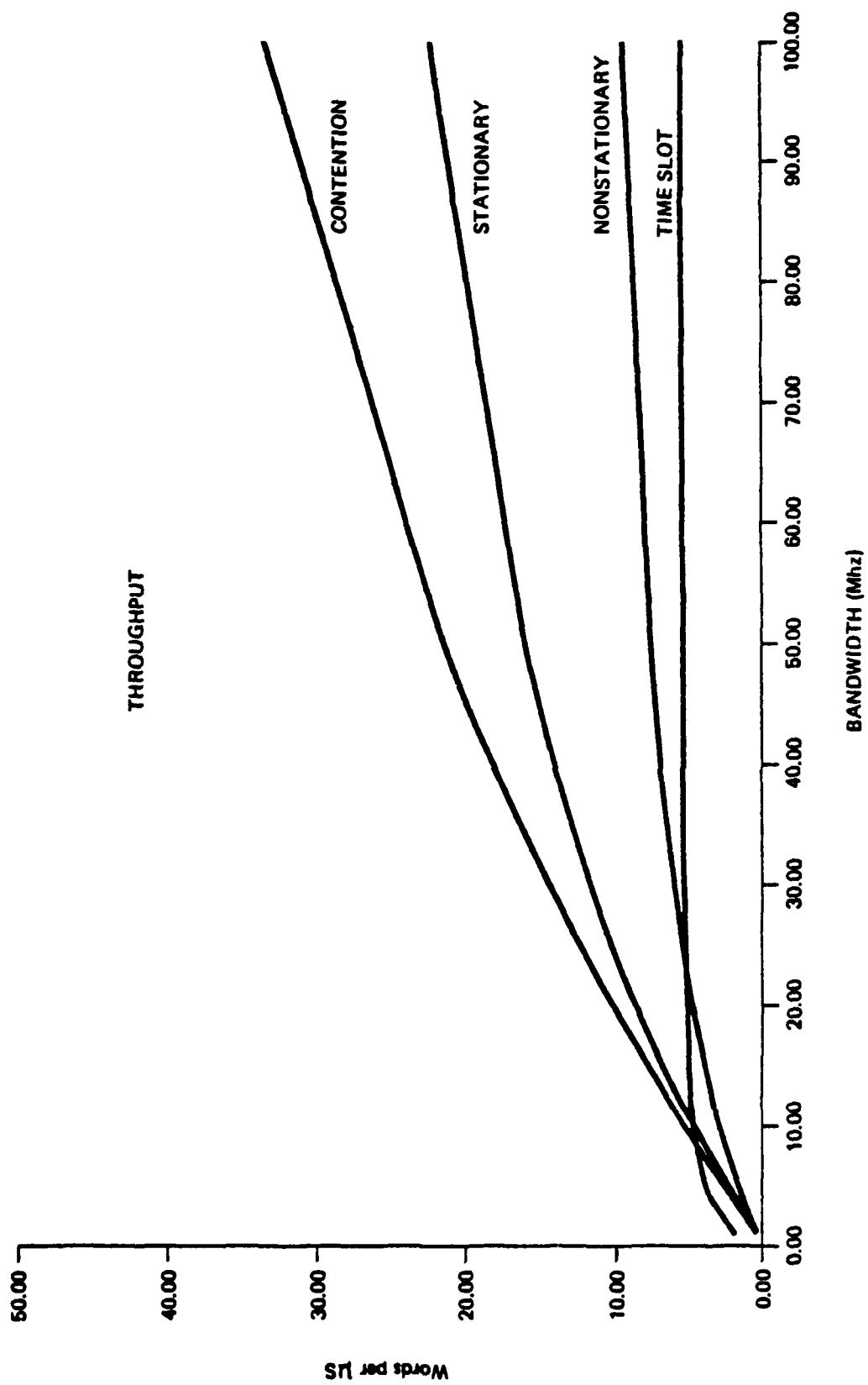


Figure 4-5. Effect of Bandwidth on Throughput

$$\text{Throughput} = \frac{\frac{16}{20} \text{ L words}}{\sum_{i=1}^8 M_i \left[ G_i \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{20}{B} (S_i + L) \right] + \frac{4 \left( \frac{1}{2} + \frac{4}{B} \right) + 5 \left( \frac{20}{B} \right) + 125}{5}}$$

#### 4.2.2.3 Contention Multiple Access ITS (CMAITS)

The CMAITS protocol is best suited for systems that are broadcast oriented. The message mix which is being used in this analysis is nominal for a command-response system and does not accurately reflect the information handling capability of a contention system, however this mix will be used to maintain a common baseline upon which a comparative analysis can be made.

The overhead on the contention protocol includes loss due to collisions. When a collision occurs, 30 bit times are consumed before the collision is detected and the message stopped and requeued for later transmission. The probability of the collision is:

$$P_c = \frac{Q}{Q-1} \left(1 - \frac{1}{Q}\right)^Q$$

Therefore the expected overhead due to collisions is:

$$\frac{30}{B} \left( \frac{Q}{Q-1} \right) \left(1 - \frac{1}{Q}\right)^Q$$

and total average message time is:

$$\sum_{i=1}^8 M_i \left[ G_i \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{20}{B} (S_i + L) \right] + \frac{30}{B} \frac{Q}{Q-1} \left(1 - \frac{1}{Q}\right)^Q$$

The throughput of the contention system is defined as:

$$\text{CMAITS Throughput} = \frac{\frac{16}{20} \text{ L words}}{\sum_{i=1}^8 M_i \left[ G_i \left( \frac{1}{2} + \frac{4}{B} \right) + \frac{20}{B} (S_i + L) + \frac{30}{B} \left( \frac{Q}{Q-1} \right) \left(1 - \frac{1}{Q}\right)^Q \right]}$$

### 4.3 SINGLE PROCESSOR SYNCHRONOUS EXECUTIVE

#### 4.3.1 Task Purpose

The primary purpose of Task V was to develop a version of the DAIS Executive that was at least one-third smaller than the original and required less processing time. Because of the versatility and adaptability of the DAIS executive to many situations, it is a large program which requires a substantial amount of time to execute (refer to the final report for the Evaluation of DAIS Technology Applied to the Integrated Navigation System of a Tactical Transport, AFAL-TR-1061, for an evaluation of the DAIS executive). In many situations, however, much of this software is not required; hence a good deal of computer memory and time is wasted. For example, in a single processor system where the data bus traffic is synchronous, the DAIS executive is much too powerful and time consuming to be a practical choice for an executive. The development of a more compact Single Processor Synchronous Executive (SPSE) would increase the likelihood that it might be adopted for use in a new avionics system, either as part of a retrofit program or as part of a new aircraft.

#### 4.3.2 Task Approach

Work on Task V began with a series of trade studies which were performed to determine which features of the DAIS executive should be retained in the SPSE. The results of the studies were documented in the second AASMMA interim report; these results are summarized in section 4.3.6 following.

Once the executive features were chosen, the approach was to use the existing DAIS executive documentation and code as a baseline and to work toward the SPSE documentation and code. Before doing so however, one firm groundrule was established: only the DAIS System Control Procedures, Part I specification and Part II specification would change. All other DAIS documentation and standards would remain in effect. This had two important implications. First, all applications software tasks designed in accordance with the SPSE Part I specification can run under either the DAIS executive or the SPSE. Secondly, all of the DAIS support software tools such as the Partitioning, Analyzing, Linking and Editing Facility (PALEFAC), may be used without modification.

More specifically, the approach to Task V was to begin with the DAIS Executive Part I specification and the System Control Procedures and examine each section and paragraph for applicability. If a section was not applicable, it was dropped. Only minor changes were allowed on those sections that were retained. A similar procedure was applied to the Part II specification. Each module, each entry on the flowcharts, and each data item were examined for applicability. Once again, only relatively minor changes were allowed on the retained portions. Finally, the code was modified on the basis of the above changes. All changes were made while rigidly following the groundrule.

Once the design modifications were completed, the SPSE Part II specification was reviewed expressly to locate areas where changing the structure of the executive might result in a reduced memory requirement. Two principal methods for saving

memory were identified. The first method was to combine two similar but separate DAIS procedures into a single SPSE procedure. The second method was to eliminate any procedures which were called by only one other procedure and include the code inline as a "Macro" in the calling procedure. In order to preserve the original DAIS module definition whenever practical, no structure changes were made unless the memory reduction exceeded 10 words.

Following the completion of the SPSE design, the SPSE was coded, compiled, linked with the Model Avionics Program (MAP), and installed in the DARTS Laboratory for test and debug. Two separate demonstrations were conducted using the SPSE. The first was in the DARTS Laboratory and was designed to verify that the SPSE could support the requirements of a typical avionics system. The second demonstration was held at AFWAL and was designed to show that the SPSE software was transportable, that the DAIS executive-to-applications interface had been maintained, and that the SPSE could pass the DAIS Validation and Verification tests. The validated software was then delivered to AFWAL.

#### 4.3.3 Documentation Produced

A number of documents were generated in support of Task V. The complete list of these documents is provided in Table 4-1. The AFWAL Review column of table 4-1 indicates those documents which were submitted first as a draft and subsequently updated before the final document was submitted. In addition, change pages to the Part II specification were provided after the final draft was submitted.

#### 4.3.4 Support Tools

Early in the Task IV effort, a review of the code development procedure for the SPSE was made. This review uncovered several potential development problems which are described in section 4.3.4.1. It was decided that several Boeing-generated development tools could substantially aid the program development effort if applied to the SPSE coding process. Two of these tools are SUBSORT, described in section 4.3.4.2, and Universal Source Files, described in section 4.3.4.3. A third Boeing tool, used to process Universal Source Files is INGEUNS, which is described in section 4.3.4.4.

Table 4-1 TASK V DOCUMENTATION

<u>Document</u>	<u>AFWAL Review</u>
SPSE Interim Report	No
SPSE System Control Procedures	Yes
Computer Program Design Specification for the SPSE, Part I	Yes
Demonstration and Test Plan for the SPSE	Yes
Demonstration and Test Procedures for the SPSE, Group I	Yes
Demonstration and Test Procedures for the SPSE, Group II	Yes
PDR Agenda	No
PDR Minutes	No
Computer Program Product Specification for SPSE, Volume I: Local Executive	Yes
Computer Program Product Specification for SPSE, Volume II: Bus Control	Yes
CDR Agenda	No
CDR Minutes	No
SPSE Functional Test Report, Group I (DARTS Laboratory)	No
SPSE Functional Test Report, Group II (AFWAL Laboratory)	No

#### 4.3.4.1 Potential Development Problems

At first glance, the job of modifying an already existing computer program to eliminate certain functional capabilities would seem to be almost trivial. However, this is not the case. Major functional capabilities of the DAIS executive, such as asynchronous bus communication, are not neatly isolated in one or two modules, but are spread throughout the program. To cleanly remove this feature requires modifications to most of the original program modules as well as to the data base. Because the program changes are so widespread, many potential problem areas exist during program development. For purposes of discussion here, the problem areas are broken down into four types:

- 1) Transcription errors
- 2) Design integrity
- 3) Loose ends
- 4) Laboratory debug

Each of these is discussed briefly below.

##### Transcription Errors

Transcription errors are introduced at the earliest stages of coding. Because the new software program is being derived from a previously existing program, every attempt is made to save as much of that old program as possible. Technical aides are assigned the task of extracting the desired portions of the original code and putting it into new source files. Since typing in all of this source from scratch is impractical, the usual approach is to copy the original file and delete unwanted lines. The initial output of this task is generally satisfactory, with relatively few errors.

If this were the only time that the original program files were referenced, this stage of program development would present few problems. Often, however, refinements in the design will require modifications during the development stage. This often entails returning to those original files to either copy or delete more material. It is during this process that one or more lines of code can be either incorrectly added or deleted, resulting in a transcription error that is hard to detect.

##### Design Integrity

In order to retain the desirable features of a program that is being modified, it is necessary to maintain the design integrity of that program. To help insure the design integrity of the SPSE, it was a requirement that all applications tasks executing under this executive could execute under the original DAIS executive as well. Since the operation of the DAIS executive is controlled by a data base which is built offline, any changes to that data base must be carefully controlled. No changes can be made unless the impact can be assessed for all modules referencing the modified data. To miss checking any of these modules introduces a risk that the integrity of the data base might be lost.

### Loose Ends

After the initial version of the new program (in this case, the SPSE) is completed, it is possible that the design changes to the original program may have left a number of "loose ends." These loose ends can take many forms, but the end result is that memory and processing time are being used unnecessarily. For example, in the original program, the designers may have avoided repetitious coding of an algorithm used in several different areas of the program by defining a separate procedure. After the modifications are completed, only one call to this procedure remains. Memory and overhead can be reduced by rewriting this procedure as inline code. As another example, one routine may be responsible for calculating a variable used by another routine. If, after the modifications, the second routine no longer needs that data, the calculation should be removed from the first routine. "Loose ends" will not prevent a program from running, but they can add substantially to the program's overhead.

### Laboratory Debug

The last major problem area is encountered while trying to debug a problem in the laboratory. This problem is really the result of uncertainty over whether the three previously described sources of error have been eliminated. If the debugging programmer has no faith that the original code was transcribed properly or that data base integrity has been maintained, much of the debugging phase will be spent in tracing the derivation of the code, rather than in attacking the observed problem.

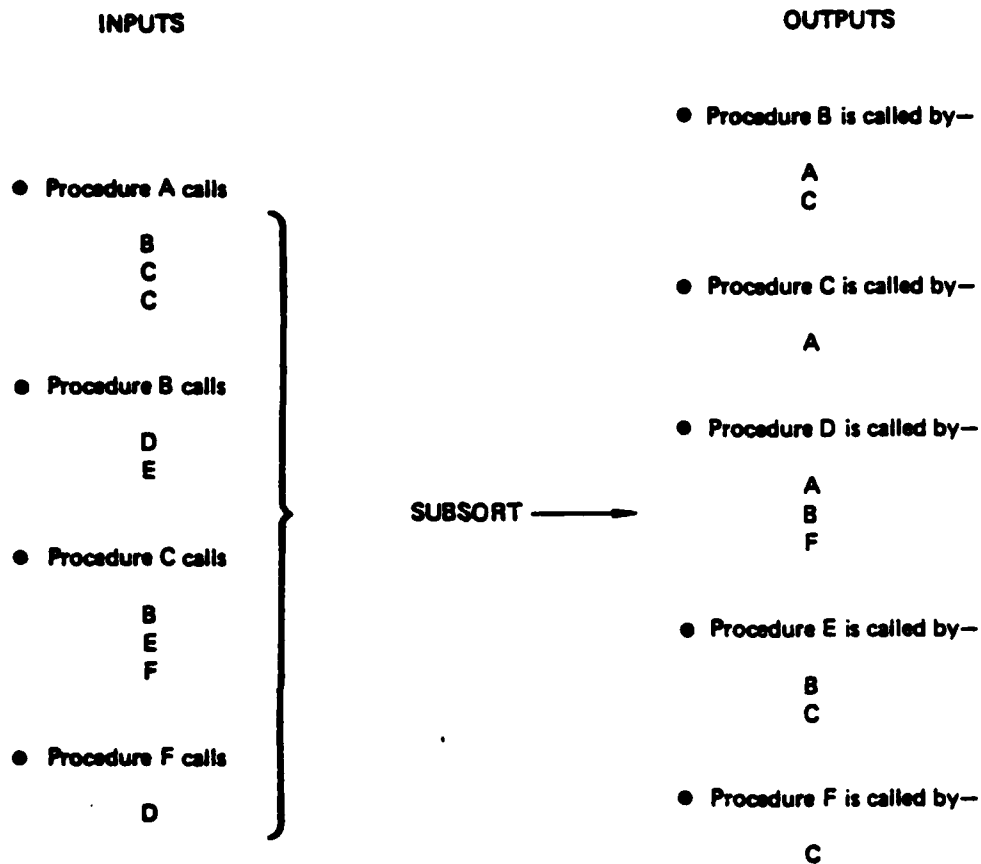
These four problem areas exist whenever one program is being derived from another. It was the desire to reduce the impact of these potential problem areas during development of the SPSE that led to the use of the Boeing-generated development tools: SUBSORT, Universal Source Files, and INGEUNS.

#### 4.3.4.2 SUBSORT

SUBSORT takes a list of all parameters used in one routine, compares it with similar lists for all other routines, and then generates lists of all routines which reference a single parameter. An example of SUBSORT inputs and outputs is shown in figure 4-6. SUBSORT can be used either to generate a list of routines which call each procedure in a computer program, or it can generate a list of all routines that reference any given parameter.

The inputs to SUBSORT can be readily generated by using the Computer Program Development Specification paragraphs on linkages and data which are provided for every function. A more direct (and more accurate approach) would be to use the listings generated by compilation of the source code as the basis for SUBSORT inputs.

The SUBSORT outputs were the primary means of finding the loose ends in the SPSE. Several variables which were being maintained in many different routines, but which were never actually used, were found and eliminated in this manner. In addition, several decisions to combine two or more routines into a single routine were influenced by the data produced by this program.



*Figure 4-6. SUBSORT Inputs and Outputs*



The SUBSORT program was also an aid in maintaining design integrity. The data compiled by SUBSORT made it much easier to locate places where changes to the DAIS executive data base could modify or eliminate information being processed in another routine.

#### 4.3.4.3 Universal Source Files

Universal Source Files (USF) can be used to maintain the concept of a family of DAIS executives. A USF contains the source code for every member of the executive family which makes use of the corresponding software module. For example, a USF for the initialization routine would contain every line of source code that is needed to generate the specific initialization procedure appropriate for a single member of the executive family. Each line of code in the USF is marked to indicate in which family member it is used.

A USF is created by executing the USF software program on a source file from the parent executive. Each line of the original file is padded with five leading blank spaces. Each of these spaces corresponds to one member of the executive family. If the line of source code is not used in one version of the executive, the corresponding space is filled with an X. If a line of code needs to be added to support a new version of the executive, all of the spaces except the one corresponding to that version are X-ed out. Although the USF's used to develop the SPSE can support only five versions of the executive, this limitation is arbitrary; the same technique can be used to support any number of versions.

The advantages of developing USFs for a family of software programs are numerous. A complete software package for all versions of the program is contained in one configuration. A user can tell at a glance which sections of a program vary substantially from one version to another and which remain fixed. A degree of configuration management is introduced to software development; by demanding that all coding changes be made in a USF, a software manager can ensure that all coding changes can be easily reviewed by system designers. There is never any question about whether a section of code is new or whether it has been revised. Quality control for code can now begin with code generation rather than with debug and test.

#### 4.3.4.4 INGEUNS

Since the USF itself cannot be compiled, another development tool is needed to extract from the USF the source code desired for one version of the executive. This tool is the INTERpreter for Generating Executives from UNiversal Source files (INGEUNS). The input to INGEUNS specifies which USF file is to be processed and which version of the executive is desired. The output is a compile-ready source file for the selected executive. An example of how INGEUNS might be used to generate individual source files from a USF is shown in figure 4-7.

By using INGEUNS in combination with the USFs, many potential development problems can be controlled. Since all versions of the software are contained in one configuration, transcription errors, while not eliminated, become very easy to

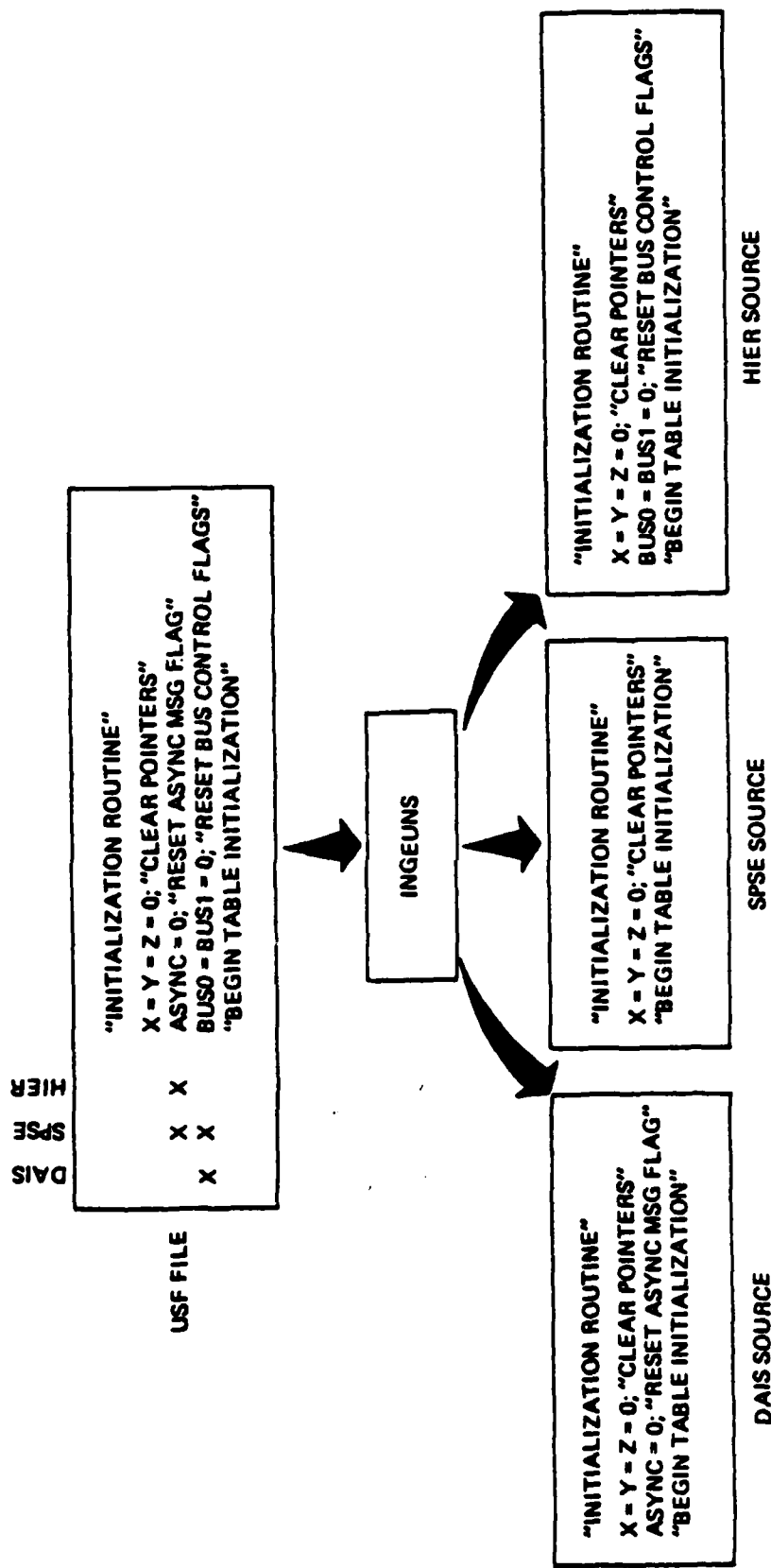


Figure 4-7. Generating Individual Source Files From a USF

track down and correct. Design integrity is assured since the basic framework of the software program remains intact in the USF. The task of debugging and testing the latest version of the program is greatly simplified because laboratory personnel can concentrate on just the areas of code that have changed. A glance at the USF is sufficient to determine if the code in question is new, modified, or unchanged from another, previously verified, member of the family.

While the use of USFs and INGEUNS was very beneficial in developing the SPSE, these development tools were not entirely without cost. The use of USFs added significantly to the software storage requirements for the development of the system. For the SPSE, a reasonable working estimate is 65% for the overhead of building and saving USFs. The USF listings must also be maintained in reference documents. Finally, the technical aides responsible for software coding needed additional training. The natural inclination of a technical aide to bypass updating the USF and make changes directly in the source file must be countered. At the very beginning of the software project, rigid procedures were established to ensure that all software modifications were done in the USF. The benefits of using USFs were so great, however, that these additional development costs became insignificant.

#### 4.3.5 Description of SPSE

Using the software tools described in the previous section, the SPSE has been built, debugged and delivered to AFWAL. By using the development tools described above, laboratory debugging was completed in two weeks.

The features of the original DAIS executive and the SPSE are compared in table 3-1. Although many of the original executive features have been eliminated from the SPSE, the basic framework of the DAIS executive has been retained unchanged. Any applications task which will run under the SPSE will also run under the DAIS executive; that is, the executive-to-applications interface has not been changed. All other DAIS standards were maintained in the SPSE, as well. The SPSE is coded in J73/I, operates in a DAIS processor using the standard instruction set, and communicates with remote terminals on a MIL-STD-1553A bus. All of these standards are shared with the parent DAIS executive.

The primary result of developing the SPSE has been the development of the second member of a family of avionics executives. The DAIS executive is well suited to large multi-processor systems which need to operate in an asynchronous environment. The SPSE is intended for use in small single processor systems which cannot afford the memory requirement of the DAIS executive.

#### 4.3.6 Comparison of DAIS Executive and SPSE

Many of the changes in the SPSE code involved removing a capability from the DAIS executive. (The other changes, described in section 4.3.7, were made to reduce the memory requirement by implementing a capability in a different manner). This section provides a list of the major DAIS executive features that were eliminated from the SPSE; the reasons for deleting each item are briefly described.

Asynchronous Bus Communications. This is the most costly single feature of the DAIS executive. Not only does it require several thousand words of memory, but the associated processing time is substantial. According to the evaluation of the DAIS executive documented in AFAL-TR-1061, the typical asynchronous bus communication requires 1.5 ms to service; the worst case is a trigger which requires nearly 3.0 ms to service. (These times are the combined service times required for the executive in the originating processor and the executive in the receiving processor.)

Many systems can be designed to handle all processing requirements with only synchronous bus transmissions. The original motivation for building an SPSE was to develop a smaller executive which would support these synchronous systems. The elimination of asynchronous bus communications was the only predetermined modification to the DAIS executive.

Multiple Processor System. Since the impetus for reducing the size of the DAIS executive was the desire to tailor it for use in small systems, elimination of multiple processor capability was a prime consideration in the new executive. The only issue was whether the new executive could be designed in a manner that permitted the easy inclusion of multi-processor capability in a synchronous system. A trade study showed that the multi-processor feature was closely tied to the availability of asynchronous bus communications. A synchronous system that supported multiple processors could not be designed without substantially altering the structure of the executive. Even if this change were made, the new executive would be only slightly smaller than the original DAIS executive. For these reasons, multi-processor capability was excluded in the new executive.

Broadcast. The broadcast feature is used either in a multiple processor system or for asynchronous updates to multiple RTs. Since the SPSE can be used only in a single processor system which does not support asynchronous bus transmissions, broadcast was eliminated.

Trigger. The trigger capability is inherently asynchronous. Even in a system which supports asynchronous transmissions, the additional 3 ms of executive processing time is very costly overhead at a time when critical applications processing might be required.

Forced Read. This feature is also asynchronous.

Terminate. The terminate statement allows a controller to put an applications task into an uninvoked, inactive state. Such an action is usually based on the receipt of an asynchronous event. In a single processor synchronous system, few events are expected to occur asynchronously. If the terminate option must be provided, it can be simulated by cancelling and then immediately rescheduling the task. The benefit of including the terminate feature was judged insufficient to justify the memory required to support it.

Time Wait. A time wait allows a task to suspend itself for a fixed period of time. This feature was judged to be of little value in a single processor synchronous system. As with the terminate feature, the benefit of including the time wait feature was judged insufficient to justify the memory required to support it. In contrast to the terminate, however, there is no alternate means of simulating a time wait other than replacing it with an event wait which would be satisfied upon receiving a signal from a task which is activated periodically

and generates a signal when a counter reaches a certain value. It is possible that this cumbersome method might cause certain applications programs to add more memory to a system than was saved by eliminating time waits. In that case, this decision may have to be reviewed.

Sensitive Terminal. By designating a terminal to be sensitive, the executive performs a careful retry whenever a transmission error is detected. The need for such a feature is minimal in a synchronous system.

SIL Done Event. The principal use of a SIL Done Event is to notify the system when asynchronous bus activity will not affect synchronous bus activity. Other uses of the SIL Done Event were judged insufficient to justify the memory requirement for the option.

Bit and Word Masking. Terminals presently available to system designers do not support this option and MIL-STD-1553B precludes the use of this option. Therefore, software support does not seem warranted.

Nonstandard Device Handler. Nonstandard devices are not expected in systems which might use the SPSE.

#### 4.3.7 Modifications to DAIS Executive Structure

In addition to removing features from the DAIS executive, the DAIS executive structure was modified in the SPSE whenever substantial memory savings were possible. The structure changes were of two forms:

- (a) Combining two similar routines into a single routine, and
- (b) Rewriting a procedure called from only one location as inline code.

The first structure change was followed for the signal, read, and write routines. Explicitly, the DAIS executive normal mode signal routine and privileged mode signal routine were combined into a single SPSE signal routine. The DAIS executive normal mode read routine and privileged mode read routine were combined into a single SPSE read routine. The DAIS executive normal mode write routine, privileged mode write routine, and local copy override routine were combined into a single SPSE write routine. The seven original DAIS executive routines occupied 798 words of memory. The three SPSE routines require only 344 words of memory for a savings of 454 words.

The second structure change was followed for four DAIS executive routines. The task termination routine was made inline code in the cancel routine, the task schedule routine was placed in the schedule routine, and the minor cycle setup routine and the dispatcher were placed in the local executive controller. These modifications saved almost 100 words of memory.

#### 4.3.8 Modifications to the Data Base Structure

In the quest to minimize the SPSE size, the data base was modified whenever memory savings were possible. Changes were made to both the internal executive data and the external PALEFAC Mission Data (PMD) files.

Changes to internal data involved removing items no longer needed by the executive and reducing the size of queues. The run queue was reduced from ten entries to six and the input queue was reduced from twelve entries to six.

Several data items defined in the PMD files are no longer used in the SPSE. Since no PALEFAC modifications were undertaken in the Task V, any changes made to the PMD files must be handled by editing the PMD files produced for any applications program. Table 4-2 lists the tables and items that must be deleted from the PMD files when linking them to the SPSE. These deletions save a total of 43 words.

Two additional changes in the PMD were made to support the laboratory testing of the SPSE. The first was adding a halt to the idle polling list to institute single pass idle polling. The second was adding an RT to the minor cycle polling list to synchronize the simulation system with the SPSE.

#### 4.3.9 Demonstration Results

Once the coding and debug phases were completed for the SPSE, the executive was demonstrated at both Boeing and AFWAL. The Boeing demonstration was conducted in the DARTS facility on 5 March 1980. The AFWAL demonstration was held at WPAFB on 2-4 April 1980.

The DARTS demonstration consisted of monitoring the processing which occurred in a single load module while MAP was requesting services from the SPSE. The demonstration was a complete success with every test showing that the executive was working properly.

The AFWAL demonstration was in four parts with each part requiring a different load module. The first load module was a re-creation of the one used in the DARTS lab. The second was the MAP software linked to a single processor version of the DAIS executive. The third was the MAP software partitioned for a two processor system and linked to a two processor version of the DAIS executive. The fourth was a modified version of the DAIS Validation and Verification (V&V) program linked to the SPSE.

The AFWAL demonstration was hampered by technical difficulties which prevented the use of the Bus Monitor Unit (BMU) for recording bus traffic. To compensate for the problem, the Universal Remote Terminal (URT) was used to display a limited set of bus messages and count the number of times they were transmitted.

Two software problems were uncovered during the AFWAL demonstration. The first problem was the failure of the two processor MAP system to activate tasks in the remote processor. Since MAP functioned properly in a one processor system linked with either the SPSE or the DAIS executive, the failure was presumed to be tied to either a data base problem or to the remote executive. For this reason, the failure was determined to have no effect on the evaluation of the SPSE or MAP software.

Table 4-2 PMD DELETIONS IN SPSE

<u>PMD File</u>	<u>DATA Type</u>	<u>Name</u>
PMD020	ITEM	P3MAXBUSY
PMD020	TABLE	P3ASYH
PMD020	TABLE	P3MRDT
PMD020	ITEM	P3MRDN
PMD020	TABLE	P3MIST
PMD020	TABLE	P3MINK
PMD029	ITEM	P3SDEO
PMD029	ITEM	P3LSYN
PMD029	TABLE	P3SYNP
PMD029	TABLE	P3SYNX
PMD029	ITEM	P3NP
PMD029	TABLE	P3TOAD
PMD029	ITEM	P3SNAK

The second software problem was uncovered while trying to run the SPSE with the V+V program. The SCADU was used to determine that the problem was in the event handling routine of the SPSE. An SPSE design decision to combine the task activation event handling routine with the event handling routine was in error. The routines were immediately recoded in their original form and a new SPSE/V&V load module was created. This load module worked as expected and the SPSE passed V&V.

The AFWAL demonstration successfully verified the SPSE code, the transportability of the software, and the adherence of the SPSE to the DAIS executive-to-applications interface standard.

#### 4.3.10 SPSE Performance

The Boeing and AFWAL demonstrations verified that the SPSE code satisfied the design requirements. All features of the SPSE described in the Part I and Part II specifications have been supplied in the delivered software.

The delivered size of the SPSE is 5564 words. This compares to 12636 words for the DAIS executive. Figure 4-8 shows how the estimated size of the SPSE varied as the design stages progressed. The SPSE has the unusual distinction for a software program of getting smaller during all phases of the effort.

Only crude timing figures can be provided for the SPSE. Two data points are available. When MAP is run under both the DAIS executive and the SPSE, the SPSE system runs approximately 5% faster. When identical versions of the V&V are run under both the DAIS executive and the SPSE, the SPSE system is 35 to 40% faster. The primary difference in the two applications programs is that MAP makes relatively few executive service requests while the V&V does very little except request executive services. The implication is the SPSE overhead was reduced only slightly for periodic services such as minor cycle setup, but that major improvements were achieved for asynchronous services such as signals and writes.



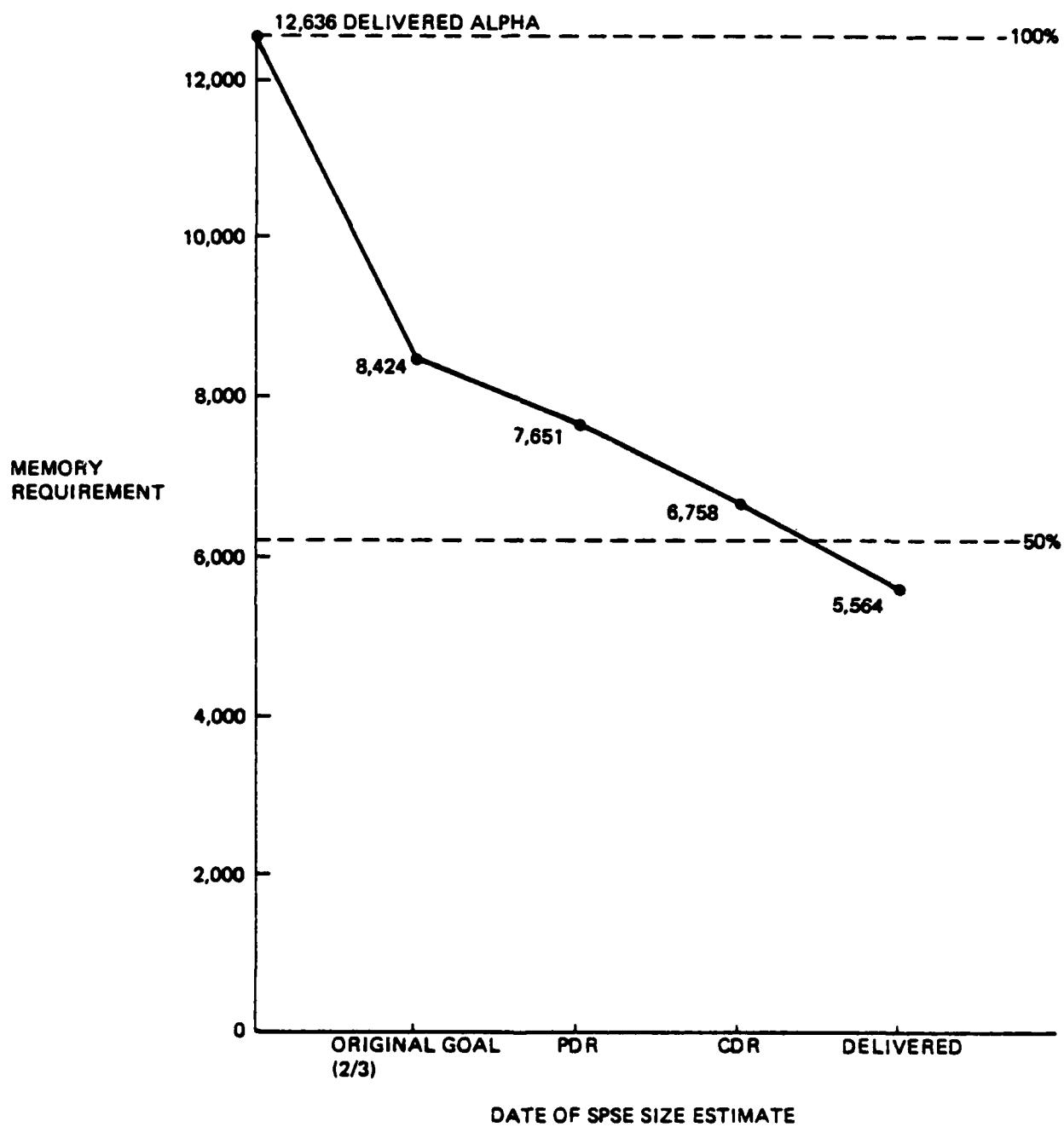


Figure 4-8. SPSE Memory Requirement Versus Time

## 5.0 RECOMMENDATIONS

The AASMMA program has provided a strong engineering baseline from which to support further work on executive software for avionic applications. The two principal achievements of AASMMA have been to define the basic approach to developing an architecture for multimission applications and to produce a compact tested version of the DAIS executive, the Single Processor Synchronous Executive (SPSE). The logical extension of this effort would be to restructure the SPSE to support multimission applications. The job of restructuring the SPSE can be conveniently broken into two tasks: the first task would be to convert the SPSE to an operational flight program (OFF) executive; the second task would take the SPSE OFF as a baseline and develop a hierarchical executive based on a stationary master architecture where all bus communications are synchronous.

The first task of converting the SPSE into an OFF would consist of several subtasks:

1. Convert from MIL-STD-1553A protocol to MIL-STD-1553B protocol.
2. Convert to a MIL-STD-1750 machine (i.e., convert from the AN/AYK-15 to the AN/AYK-15A).
3. Convert to the MIL-STD-1589A Higher Order Language (i.e., convert from J73/I to J73).
4. Enhance the error handling capabilities of the SPSE.
5. Implement a startup/loader function.
6. Provide for backup bus control.
7. Enhance performance by reducing size of the data base and decreasing the overhead.

The first three subtasks update the SPSE to conform with the latest applicable military standards. The next three subtasks upgrade the SPSE from a strictly laboratory tool to a program capable of being an OFF. The last subtask is designed to enhance the attractiveness of the SPSE to designers evaluating executive options for new systems.

The second part of restructuring the SPSE would be to take the SPSE produced in Task I, study the best means of adapting it to a hierarchical architecture, as produced in Task IV of the current study, incorporate the study results in a design specification, develop a product specification, code the executive and test it. The principal problems to be solved before completing the hierarchical design specification would be to:

1. Define the procedure for interfacing with two buses in a single processor.
2. Isolate the two bus control modules to prevent mutual interference.
3. Determine which executive routines can service the needs of both buses as a single module.

4. Develop a means of transferring data from one bus to the other.
5. Determine how applications tasks in the processor can be linked to either bus, as the system designer chooses.
6. Establish a design which will support multiple failure modes. As an example, a processor which was initially serving as the master processor on one bus and the remote processor on the other bus should be capable of the following recovery modes:
  - a. Function as the master processor on both buses if the remote has a monitor function which takes over when the master processor on that bus fails.
  - b. Function as a single sub system if either one of the buses completely fails.
7. Design the support software necessary to automatically create the data base for a hierarchical system.

Once these problems were solved, the resulting hierarchical system could be developed into an OFP.

Upon completion of these tasks, AFWAL would have three different avionics executives:

1. DAIS - for large avionic systems requiring multiple processors and asynchronous communications on a single bus.
2. SPSE - for less complex avionic systems operating in one mission computer and requiring only synchronous communications on a single bus.
3. Hierarchical - for distributed avionic systems which use several buses to achieve multimission capability.

Each of these executives would conform to the requirements of the DAIS executive-to-applications interface standards and all applicable military standards (MIL-STD-1553B, MIL-STD-1750, and MIL-STD-1589A). The concept of a standardized approach to the definition and development of an avionics executive would then be firmly established.

In order to anticipate the requirements for avionic systems in the late 1980's, two additional tasks should be undertaken. The first task is to convert one of the three executives into the ADA language. This task would uncover potential problems in applying ADA to the existing executive design and would also test the validity of the executive-to-applications interface standard when used with another language. The second task would be to target one of the executives to microprocessors. This task would highlight any difficulties in using microprocessors to control bus communications. The SPSE is the preferred executive for use in both of these tasks since it is the least complicated of the three alternatives.

APPENDIX A  
RATIONALE FOR STATIONARY MASTER INFORMATION TRANSFER SYSTEM  
CONTROL PROCEDURES

The stationary master information transfer system (SMITS) was designed using DAIS as the baseline. SMITS is significantly different from DAIS in four main areas.

- (1) DAIS has a single level bus network; SMITS has a hierarchical (multi-level) bus network.
- (2) DAIS uses federated minicomputers; SMITS uses distributed microcomputers.
- (3) DAIS employs its own version of MIL-STD-1553A protocol; SMITS employs MIL-STD-1553B protocol.
- (4) DAIS uses the DAIS-specific BCIU; SMITS uses the AFWAL/ADH sponsored BIU chipset.

The SMITS control procedures document was written using the DAIS control procedures as a starting point. This appendix documents the rationale for changes to the DAIS control procedures and for the new and redesigned procedures which make up the SMITS control procedures.

The paragraph numbering in this appendix corresponds directly with the paragraph numbering of the SMITS control procedures. For example, the rationale for paragraph 4.3.1 of the SMITS control procedures is contained in paragraph 4.3.1 of this appendix.

### 3.0 SMITS SYSTEM OVERVIEW

This paragraph was revised to describe the SMITS architecture which consists of distributed microprocessors arranged in a hierarchical network. The SMITS is a MIL-STD-1553B multiplex system.

#### 3.1 GENERAL DESCRIPTION

This paragraph describes the SMITS network of microprocessor/BIU's arranged in a hierarchical fashion.

#### 3.2 CORE ELEMENTS

A typical stationary master hierarchical system architecture is presented, which is markedly different from DAIS because of the multi-level network.

##### 3.2.1 Bus Interface Unit (BIU)

In this paragraph, the AFWAL/ADH BIU chipset is specified, instead of the DAIS BCIU.

##### 3.2.2 Remote Terminal (RT)

This paragraph describes stand-alone and embedded RT's applicable to the microprocessor based subsystems which form SMITS.

##### 3.2.3 Processors

This paragraph suggests the use of commercial microprocessors or MIL-STD-1750 military machines.

##### 3.2.4 Mission Software

This paragraph was changed to delete the reference to OFP and OTP being selectable by the startup/loader program. In the SMITS, the OFP and OTP would be different mass memory loads and the startup/loader would load the system with whichever program (OFP or OTP) was in the mass memory.

#### 3.2.4.1 Executive Software

The master and local executive descriptions are similar to the DAIS equivalent except for the explanation concerning the portions of each executive which will be in various processors (master, remote, monitor or interbus). These explanations are now based on the hierarchical network function of the particular processor. Memory protection and local error handling functions were added to the local executive. Many of the future avionic processors will have extended memory mapping (see MIL-STD-1750A) and memory protection.

#### 3.2.4.2 OFP Application Software

The only change was to delete the subsystem status monitor as a separate functional category and to make it a part of configuration management. A function of configuration management is to maintain an account of errors for each specific system. The subsystem status monitor will remain as a "separate" function, but is included in configuration management.

#### 3.2.4.3 OTP Applications Software

The OTP is hardware configuration dependent.

#### 3.2.5 System Mass Memory

The main change in this top level description is that the mass memory will be accessible from two bus levels (a global and a local level).

#### 3.2.6 Processor Control Panel

The panel was changed mainly to eliminate the separate power enable buttons for each bus side and processor. As the complexity of the distributed system grows in terms of number of processors and levels of hierarchy, it is obvious that the pilot cannot be required to monitor the performance of each bus, bus side, and processor. The software itself will make most of the configuration decisions and implement those decisions automatically, without the previous tedious and error-prone procedure of "advising" the pilot of status and waiting for his manual inputs. As the distributed architecture evolves, more redundant capabilities will be available. To make full use of these capabilities will require automatic reconfiguration with minimal pilot interaction. The power switches are normally circuit breakers which are part of the overall avionic system configuration.

Another change was to eliminate the "ground/inflight" switch on the PCP. In its place, a "weight-on-gear" switch will be provided to reflect the vehicle position automatically. Again, this was an effort to free the pilot from unnecessary manual actions.

The "START" switch was changed to "RESTART." The pilot powers up the processors via the "POWER" button, then presses the "LOAD" button. The processors then begin execution as soon as they are loaded and have established a good configuration.

The "OTP/OFP" switch was eliminated because it was felt that the OTP would not be available from the OFP system mass memory. If the OTP were to be run, it would most likely be invoked from a separate mass memory "cartridge" via the LOAD button or to be loaded through the automatic ground equipment (AGE) interface. An alternative mechanism to load the OTP if the OFP is operational, is to load the OTP into high-memory via the mass memory interface, and for the executive to invoke that OTP as another "task".

## 4.0 SYSTEM CONTROL PROCEDURES

This introductory paragraph lists the system operational modes, which are very similar to DAIS. The procedural differences between SMITS and DAIS show up in the detailed rationale which follows.

### 4.1 SYSTEM STARTUP/RESTART OPERATIONS

The major change to this section was to remove the system configuration from the control of the pilot. As explained in the rationale for the changes made to the PCP, it was felt that the pilot must be freed from the responsibility of monitoring the performance of a complex distributed processing system. The operational software itself must make and implement decisions regarding the configuration. The pilot will be kept informed of these decisions, but will not normally be required to take part in them.

#### 4.1.1 Normal System Startup

##### 4.1.1.1 Startup Self Testing

The startup self testing routine resides in each processor's memory. It is invoked when the LOAD button is pressed as well as at power up including after a power transient. This change was made to simplify the entire startup/restart process by limiting the number of alternate paths. The processor always invokes its ROM to start or to do a cold restart. Once a technology such as shadow memory becomes available, the loader and self test could be written over during the course of normal operation.

##### 4.1.1.2 Bus Control Arbitration

The interbus processor is required to interface to the startup/restart operations of both bus levels. The bus control arbitration procedure was also changed so that if the bus controller cannot obtain a verified system loader in its own memory, it will simply allow another processor to take control. This change was made to simplify this procedure by eliminating the possibility of having the bus controller and system loader in different processors.

##### 4.1.1.3 Configuration Identification

The main changes were in response to the PCP changes explained earlier. An additional change was made to respond to a failure of an attempted warm start. In this case the configuration identification procedure will proceed to the software verification/loading process instead of attempting to warm start again.



This procedure is done in order to avoid a non-productive infinite loop between the warm start and configuration identification procedures. If a warm start fails for an inflight restart, a cold start should be attempted unless the mission is in a critical phase.

#### 4.1.1.4 Software Verification/Loading

If the bus controller cannot verify that it has a correct load module after three attempts to load it from mass memory, it will relinquish control of the bus. This change acknowledges the fact that a faulty processor should not control the bus. After two attempts to verify a configuration, the warm start procedure is invoked to attempt to identify and use an alternate set of load modules. This procedure avoids an infinite loop of attempting to verify a particular set of load modules and processors.

#### 4.1.2 System Restart

The design of the system restart/warm start is a function that is controlled by the system designer and must reflect the actual needs of his aircraft, as is the case with all of the other error handling and recovery software. These algorithms show one particular mechanism to eliminate pilot interaction in this hierarchical system.

##### 4.1.2.1 System Warm Start

The major change is in the action to be taken should all attempts to warm start all available load sets fail. To illuminate a light to alert the pilot and to attempt a warm start again is a hopeless gesture. Instead, the action taken hinges on whether the current mission phase is critical. If it is and a monitor is available, the best course of action is to direct the monitor to take control. This monitor switch provides the fastest recovery possible. If the current mission phase is not critical, then the best course of action after a warm start failure is to force software verification/loading and cold initialization to take place, as if the RESTART button were pressed while on the ground.

##### 4.1.2.2 Pilot Initiated Restarts

These procedures were changed to reflect the modified PCP and to make them as simple as possible. "Background reloading" was eliminated as being too complex to implement with much success. With the software making most reconfiguration decisions and keeping the pilot informed of the system status, it was felt that the RESTART and LOAD buttons would only rarely be used and could best be implemented by simply causing each processor to invoke its ROM. In this way no software is duplicated between the startup and restart procedures since the software is common to both.

#### 4.1. System Mass Memory Protocol

Changes were made from DAIS to give the mass memory controller more processing capability to simplify the master executive. Under the DAIS concept, the master executive is required to search the mass memory directory with READ commands to find the desired file. SMITS procedures specify that the file name be passed to the mass memory controller and then it searches the directory to find the file to perform the required operations.

The additional processing capability of the mass memory controller is also used to make mass memory access faster. To do this, advantage was taken of all of the subaddresses available in the mass memory controller. A file longer than 32 words can be accessed by a single command using successive subaddresses to read or write the information. The mass memory will read or write up to twenty-nine 32 word messages as a single transaction.

Also, to make the mass memory access faster, the executive preprocessor should be used to build synchronous bus instruction lists for each mass memory read or write required during the mission to eliminate the asynchronous method used by DAIS.

The communication with a mass memory will also have to be potentially used at very low levels of hierarchy on buses that do not interface directly with a mass memory unit (MMU). In these instances the higher level multi-bus interface will have to simulate a MMU during loading, so that data can be acquired and then passed through to next level.

Note that polling of the device can be done periodically following the request to read/write. The asynchronous request vector can be set when the MMU grants access and is ready to communicate. It can do a second asynchronous request vector when the I/O is complete.

Decentralized processing allows for the removal of most of the mass memory control from the executive and move it to the mass memory controller micro-processor for the following reasons:

- (1) Mass memory unit (MMU) control for errors are best handled at the MMU, especially those concerned with characteristics of the device.
- (2) The master should be concerned with only assuring that messages get transmitted between requesting PE and the MMU. Auto-retries from transmission problems should be handled via the normal procedures. The object is to create a MMU which appears to be an RT in all transmission respects. Consequently 29 subaddresses are allocated for data to be input or output. This large number of subaddresses permits sequential and uninterrupted transfer of data of up to 928-16 bit words and remain within the MIL-STD-1553B format.
- (3) Asynchronous messages, requests, and interrupts should be minimized. The communication is considerably simplified with the control technique defined.

## 4.2 PREFLIGHT/POST FLIGHT TEST OPERATIONS

This procedure is the same as in DAIS.

## 4.3 NORMAL SYSTEM OPERATIONS

The description of paragraph 4.3.9, Power Control, was deleted from this section. The subject of power control (i.e. recovery from power transients) was TBD in the DAIS control procedures and is not included in the SMITS control procedures.

### 4.3.1 Bus Control Operations

The text of this section was modified to refer to MIL-STD-1553B and the AFAL/ADH BIU. BCIU references were changed to conform the operation of the BIU. DAIS mode command references were changed to 1553B terms or eliminated. An explanatory sentence was added which stated that: "In general, asynchronous operations are not given priority over synchronous operations." The reason for this statement was a desire to save overhead by classifying asynchronous messages as high or low priority. Potentially there will be no high priority messages that will require the interruption of normal processing. The high priority messages would still be performed immediately, but the others would be queued into a low-priority asynchronous bus list to be performed after completion of the synchronous bus list. Message transmission during a minor cycle was changed significantly from DAIS to reflect high priority asynchronous messages interrupting the synchronous bus list, a separate low priority asynchronous bus list, a polling sequence where each terminal is only polled once, and dead bus time (except for critically timed messages) until expiration of the minor cycle to reflect the diminished emphasis on asynchronous messages. MIL-STD-1553B command, status and data words are presented. The 1553B status word was adapted for SMITS by not implementing the instrumentation bit, broadcast command received bit and dynamic bus control acceptance bit. The instrumentation bit was not implemented because it would restrict the number of usable subaddresses to 15 by forcing the first bit of the subaddress field to be 1 in all command words. The other two status bits were not used because broadcast and dynamic bus control are not implemented in the SMITS. A bit by bit definition of the instruction word format used by the BIU is also presented.

#### 4.3.1.1 Minor Cycle Synchronization

The hierarchical buses are independent with respect to minor cycle, since the subsystems on each bus will likely have different timing (cycling) requirements. Since the interbus processor must be involved with the processing of each bus to which it interfaces, it will need to respond to the minor cycle synchronization demands of both buses.

#### 4.3.1.2.1 Synchronous Bus Message Operation

This paragraph was modified to reflect the fact that only high priority asynchronous messages will interrupt the synchronous bus list. DAIS BCIU references were changed to BIU. BIU "NO-GO" and "Busy" states were used rather than BCIU "quiescent" and "pseudo-wait" states. The BIU BIT word and 1553B status word definitions caused changes to the procedure flows.

#### 4.3.1.2.1 Processor/BIU DMA Sequence (Master Mode)

This paragraph was completely rewritten based on the operation of the BIU in the master mode. A new figure depicts the operation of the BIU/processor interfaces.

#### 4.3.1.2.2 Processor/BIU DMA Sequence (Remote Mode)

This completely rewritten paragraph presents the operation of the BIU in remote mode.

#### 4.3.1.3 Asynchronous Bus Message Operations

The asynchronous operations have been almost completely redesigned for the SMITS. The three main reasons for this redesign were (1) the change from DAIS/1553A to 1553B, (2) the change from the DAIS BCIU to the AFWAL/ADH BIU and (3) the assumption that most asynchronous operations have lower priority than the SIL.

##### 4.3.1.3.1 Interprocessor Asynchronous Messages

Two possible implementations of asynchronous interprocessor operation are discussed, the first having a final handshake message. This implementation incorporates hardware characteristics of the BIU as well as 1553B protocol. The requested asynchronous operation does not cause the remote processor or terminal to advance its asynchronous receive or transmit queue. This allows message retries to be performed without first realigning the remote processor's asynchronous queue for each retry. The flow finishes with the master sending a final handshake message (to the remote's subaddress 30) which causes the remote to advance its asynchronous queue, thereby also notifying the remote's host that the asynchronous operation was successfully completed.

The second method of asynchronous interprocessor operation is one without a final handshake. Unlike the first flow, the accomplishment of the asynchronous message automatically advances the remote's asynchronous queue. This allows the next asynchronous message to request service, without the need of a final handshake to

advance the asynchronous queue. The drawback to this method is that if a message error occurs in the asynchronous operation, the remote's asynchronous queue will now be pointing beyond the message which must be retransmitted. To effect a retransmission the master must send a special "realign" message to the remote before each message retry. To transmit the "realign" message would require an interrupt to the master processor each time thereby having a significant time penalty.

#### 4.3.1.3.2 Critically Timed Asynchronous Message Operation

This operation remained similar to the DAIS equivalent.

#### 4.3.1.3.3 Remote Terminal Asynchronous Operation

This paragraph was rewritten to incorporate changes due to 1553B, the BIU and a redesign of the RT serial-digital channel operation. A subsystem sends data to its RT, and the RT sets the service request bit in its status word register and loads the subsystem's vector word in its mode data register. Based on the vector word the master decides whether to perform the operation immediately or whether to add the operation to the low priority asynchronous bus list. The message operation may involve any subaddress in both master and RT; however, if subaddress 30 is used, that unit (master or RT) will receive an asynchronous interrupt to inform the unit that the operation has been completed or that data is available. An optional handshake message to the RT subaddress 30 (and if desired from master's subaddress 30) can signal the RT to perform functions such as restarting a serial-digital channel which was locked out during the current message operation, or telling the RT to realign its asynchronous queue to retransmit a message because of a message error.

#### 4.3.1.3.4 Status Polling

This paragraph was changed: (1) to reference the BIU, (2) to reflect the addition of a low priority asynchronous bus message list after the SIL and before status polling, and (3) to change polling to be accomplished once to each terminal and the bus remain quiet (except for critically timed messages) until the beginning of the next minor cycle.

#### 4.3.2 Mode Command Operations

This section was completely rewritten to define the mode codes implemented in the SMITS. The rewrite was caused by the change to 1553B defined mode codes, and their specific implementation by the AFWAL/ADH BIU. A subset of the 1553B mode codes was chosen to provide all of the capabilities required by the SMITS. The selected mode codes are listed and defined in the control procedures. Several 1553B mode codes were not implemented in SMITS. Dynamic bus control was not implemented because there is no requirement for dynamic bus control transfer.

This was not considered a desirable means for transferring control to the monitor. Synchronize (without data word) was not implemented because all synchronization messages need the synchronize data word (provided by mode code 17) to transmit minor cycle information. Selected transmitter shutdown is unnecessary because each hierarchical level is assumed to have two buses used in an active/standby fashion which will be controlled by the transmitter shutdown mode code. The BIU instruction word has a single bit dedicated to indicate on which of two buses (A or B) a message will be transmitted. Therefore, a mode code capable of controlling more than two buses per level is unnecessary in the SMITS. Override selected transmitter shutdown was not implemented for the same reasons as was selected transmitter shutdown. The equivalent function for two buses per hierarchical level is provided by the override transmitter shutdown mode code.

The flows presenting mode command operations were modified to incorporate the SMITS mode codes and reflect BIU method of operation.

#### 4.3.3 Error/Failure Management

##### 4.3.3.1 Message Retry Classes

Changes were made from the DAIS procedures to eliminate from the executive the message retry classes that would almost never be used, yet allow the system user to implement those message retry classes required for a particular application. The executive size is thus reduced by eliminating unnecessary software. The DAIS control procedures specify six classes of retry. Class I is an auto retry and the other five are handled by the executive. The SMITS control procedures changes this to two classes of retry, the class I auto retry and a class II user defined retry.

Class II retries include user defined retries. This is a set of retries that are optionally defined based upon the needs of the functioning operational system. If the set of remote terminals and applications do not require these functions then they need not be designed into the system. The careful retry and sequential retry should be available software options, but independently available functions. Likewise problems with remote terminal retry functions and conditions under which an RT can be suspended from operation should exist as a separate module, so that either an "example one" or a system designer's one can be substituted without affecting the remainder of the executive. This independence means that a well defined interface must exist for manipulating the BIU instruction lists so that terminal configuration management can occur.

##### 4.3.3.2 Interrupt Processing (Master and Remote)

The BIU interrupt scheme is entirely different from that of the DAIS BCIU. There are no "levels of interrupt" in the BIU. Conditions which require an interrupt will cause the BIU to set bit(s) in the BIU's ISR and BIT register. The BIU scans the ISR after every bus operation and interrupts its host processor upon detecting any non-zero bit. A non-zero indication in the ISR's most important bit

(bit 11) indicates a fatal transfer error or a power-on-reset and causes the BIU to halt immediately and to interrupt. The processor must read the BIU's ISR and BIT register, and analyze them to determine how to service the interrupt. The BIU ISR and BIT register are presented and described in this section. New flows show how a BIU (master or remote) presents an interrupt to its processor, and how that processor decodes the interrupt to initiate a response.

#### 4.3.3.3 Status Word Analysis

The BIU operation dictated changes in the description of how status errors and exceptions (RSE, XSE, RSEX and XSEX) are presented to the processor via the ISR and BIT register. MIL-STD-1553B defines the status word differently than DAIS; therefore, the 1553B status word definition as implemented in SMITS was examined and a modified flow for status word analysis designed. The overall flow was simplified by deletion of the mass memory unit and station logic unit specific material of the DAIS implementation because all remotes use a common status word in 1553B.

#### 4.3.3.4 BIT Word Request and Analysis

This paragraph and its related flow were modified to incorporate the analysis of the BIU BIT word, the BIU method of operation and the use of 1553B/SMITS mode code 19 to retrieve the BIT word from a remote.

#### 4.3.3.5 Terminal Failure Analysis

This procedure was modified to present how BIU implementation in master and remote processors and remote terminals should allow terminal failure analysis. Whether or not these procedures will work in a specific BIU based bus interface is hardware implementation dependent. MIL-STD-1553B/SMITS mode codes were incorporated in this procedure.

##### 4.3.3.5.1 System Mass Memory Error/Failure Analysis

This procedure was modified because decentralized processing allows the removal of most mass memory control from the executive and placement of that control in the mass memory controller microprocessor. Mass memory unit (MMU) errors and failures are best handled at the MMU, especially those concerned with characteristics of the device. The master should only be concerned with assuring that message transmission is accomplished between requesting terminal and the MMU. Auto-retries from transmission problems should be handled via the normal procedures. The object is to create an MMU which appears to be an RT in all transmission respects.

#### 4.3.3.6 Core Element Tests

Similar to DAIS control procedures.

##### 4.3.3.6.1 Processor Tests

Similar to DAIS control procedures.

##### 4.3.3.6.2 BIU Test

The BCIU Power On Reset Test was deleted because the BIU does not perform like the BCIU when power-on-reset occurs. The procedure for BIU test will vary depending on the specific BIU/microprocessor interface implementation.

##### 4.3.3.7 "FATAL ERROR" Management

This paragraph is similar to that of DAIS. No "fatal errors" have been determined for the SMITS. Specific errors will be documented as experience with an actual system accumulates.

#### 4.3.4 RT Serial/Digital Operation

RT serial digital operation was essentially eliminated from the executive and moved into the remote processing elements to be performed directly as an I/O function of the executive. This was done because (1) MIL-STD-1553B does not directly or easily support serial digital operations and to do it in a way similar to DAIS results in contorted use of 1553B, such as using synchronize mode codes to perform unintended functions like resetting the service request bit and removing serial digital channel lockout. (2) There is no reason to insist on centralized control of serial digital operations when a microprocessor can easily handle inputs as a function of its executive input/output. The error handling of a serial digital transmission is handled locally by retrying read or write operations. Release of the serial channel can be authorized by an asynchronous message or by convention using multiple subaddresses. The purpose of serial digital operations is to capture and hold inputs until successfully read. This requirement does not restrict the communication to a single subaddress, so that the "holding" can be performed sequentially and transmitted either synchronously or asynchronously, depending on system design.



#### 4.3.5 Application Executive Services

These procedures are the same as in DAIS.

#### 4.3.6 Mission Application Tasks

The mission application tasks were modified by making the subsystem status monitor function part of configuration management.

#### 4.3.7 Configuration Management

SMITS configuration management is quite different from DAIS mainly because of the hierarchical architecture of SMITS. Configuration management is a configuration-dependent function which must be altered for each configuration of hardware and possibly of software. This function exists separately and operates independently for each bus in the hierarchical system. An interbus processor will therefore be a member of two configurations, but a configuration management function will not generally need to be aware of this fact (with the exception that the processor can serve the special function of providing I/O to another "subsystem;" that is, another bus level).

All configuration status is reported "upward" from one configuration manager to another. The status of each "subsystem" is periodically monitored by the higher level configuration managers since a change in configuration may result in a change in quality or availability of data. The structure of the configuration status and the decision as to the disposition of an altered configuration is an application dependent function.

#### 4.3.8 Monitor Management

This procedure is essentially the same as DAIS; however, emphasis is placed on minor cycle slippage as the primary bus control switchover method in SMITS.

#### 4.4 BACKUP OPERATION

The "Recovery" mode of operation was eliminated as a term, since failure of a monitor processor will be handled by the warm start procedures just as any processor failure, and having a special case called "recovery" would be misleading.

Also the definition of "Backup" mode was changed. If the master processor fails during a noncritical mission phase, the monitor will take control and reconfigure via the warm start procedures. The mode then entered is not Backup, but simply a

continuation of normal processing with a different configuration. Backup mode is entered only if the master fails during a critical mission phase. This mode is not a continuation of normal processing, but is instead a different set of software tasks residing in the monitor processor.

#### 4.5 SYSTEM RECONFIGURATION OPERATION

This section was changed due to the PCP changes. The pilot no longer powers down individual processors, so "reconfiguration" is simply an inflight restart initiated by the pilot pressing the LOAD or RESTART button.

APPENDIX B  
RATIONALE FOR NONSTATIONARY MASTER INFORMATION TRANSFER  
SYSTEM CONTROL PROCEDURES

The nonstationary master information transfer system (NSMITS) is based on the stationary master information transfer system (SMITS), which in turn, is based on the DAIS system. The system control procedures for the nonstationary master system were written using the stationary master system control procedures as the baseline. This appendix documents the rationale for changes made to the stationary master control procedures in the design of the nonstationary master.

The paragraph numbering in this appendix corresponds directly with the paragraph numbering of the nonstationary master system control procedures. For example, the rationale for paragraph 3.2.5 of the nonstationary master system control procedures is contained in paragraph 3.2.5 of this appendix.

### 3.0 NSMITS SYSTEM OVERVIEW

A nonstationary master information transfer system differs from a stationary master system in that several processors on the NSMITS bus are given their turn as master of the bus during a minor cycle. The major reason for having a nonstationary master system is to gain additional capabilities and independence for multimission applications.

Bus control transfer between master processors on the NSMITS bus occurs in a round robin sequence. A round robin protocol was chosen over a polling protocol for its simplicity and low overhead. See the rationale for section 4.3.1.1 for more details on the round robin protocol.

Two possible bus control transfer timing intervals were examined:

1. Each nonstationary master on the NSMITS bus functions as a bus controller once per minor cycle.
2. Each nonstationary master on the NSMITS bus functions as a bus controller for one complete minor cycle. Each nonstationary master would be a bus controller every N minor cycles, where N is the number of nonstationary masters.

The first of these methods was chosen because it provides more frequent access to the bus for data transmission.

### 3.1 GENERAL DESCRIPTION

In this section the concept of primary and secondary masters is introduced. In a nonstationary master system there is still the requirement that one processor be in control of the system timing and the other masters. This processor is designated the primary master. Other processors which participate in the round robin sequence of bus control transfer are called secondary masters.

### 3.2 CORE ELEMENTS

No change.

#### 3.2.1 Bus Interface Unit (BIU)

No change.

#### 3.2.2 Remote Terminal (RT)

No change.

### 3.2.3 Processors

No change.

### 3.2.4 Mission Software

No change.

#### 3.2.4.1 Executive Software

The master executive of the stationary master system was divided into a master executive and a system control executive for the nonstationary master system. This modularization was adapted to reduce the size of the executive in the secondary master processors by eliminating unneeded functions. Both the master executive and the system control executive are required in the primary master and monitor processors. The bus control executive is not required in the secondary master. Thus, the bus control executive is given the function of supporting the role of primary master.

The bus control function of the master executive was expanded to include the capability of accepting and giving up control of the data bus. This expansion is necessary to accommodate the protocol of the nonstationary master system.

#### 3.2.4.2 OFP Applications Software

In order to support multimission capability, the control procedures for the configuration function of the OFP applications software state that each of the bus controllers operate independently from the other controllers with respect to task control.

#### 3.2.4.3

No change.

### 3.2.5 System Mass Memory

The stationary master information transfer system control procedures specify the mass memory controller to be an interbus processor. In the nonstationary master system control procedures this was changed to make mass memory connected directly only to the global NSMITS bus. The reason for this is the possibility of a tree structure hierarchy of many different sub buses. If the mass memory was to be an interbus processor then it would probably be between the global bus and the core

avionics sub bus. This idea presents timing problems however. A master on the NSMITS bus could have problems finding time to access mass memory if it had to wait while the sub bus had control of it. With the mass memory connected to the global NSMITS bus only, each bus controller can access mass memory when it is master and relay mass memory requests from its sub bus at that same time. See the rationale for section 4.1.3 for details on mass memory protocol changes.

#### 3.2.6 Processor Control Panel (PCP)

No change.

#### 3.3 SUPPORT SOFTWARE

No change.

#### 4.0 SYSTEM CONTROL PROCEDURES

A bus control management function was added to the normal system operation mode. The rationale for this addition can be found in section 4.3.8.

#### 4.1 SYSTEM STARTUP/RESTART OPERATION

No change.

##### 4.1.1 Normal System Startup

A "lower bus still loading" flag was added to the startup status information of interbus processors. The rationale for this is explained in section 4.1.1.3, Configuration Identification.

##### 4.1.1.1 Startup Self Testing

No change.

##### 4.1.1.2 Bus Control Arbitration

If system loaders are not in ROM then they have to be loaded into the required processors from mass memory. The loading of system loaders must occur in a top/down order with the global bus first receiving its system loader, then interbus processor to lower levels receiving system loaders. This is necessary because a system loader is required in the higher bus levels to support a mass memory transaction by the next lower level.

##### 4.1.1.3 Configuration Identification

Configuration identification and software module loading occurs in the lowest bus levels first and proceeds up the hierarchy until the global bus is identified and loaded. This procedure allows a higher bus level to change its configuration based on the configuration of lower levels. A "lower bus still loading" flag is used by interbus processors to prohibit the higher bus level from configuring and loading until the lower level is finished and clears the flag.

#### 4.1.1.4 Software Verification/Loading

No change.

#### 4.1.1.5 Cold Initialization

No change.

#### 4.1.2 Restart

No change.

##### 4.1.2.1 Warm Start

No change.

##### 4.1.2.2 Warm Initialization

No change.

##### 4.1.2.3 Pilot Initiated Restarts

No change.

#### 4.1.3 System Mass Memory Protocol

The NSMITS mass memory is connected only to the global bus. For this reason, the SMITS mass memory operation of Release Access is deleted in the NSMITS protocol and a new operation, End of Transfer, is defined. Also the Bus Level information included in the mass memory status is deleted in the NSMITS mass memory protocol.

#### 4.2 PREFLIGHT/POST FLIGHT TEST OPERATIONS

No change.



### 4.3 NORMAL SYSTEM OPERATION

The following operating modes were changed or added in the change from stationary master to nonstationary master:

- Bus Control Transfer
- Minor Cycle Synchronization
- Mode Command Operations
- Bus Control Management

The rationale for these changes or additions is contained in the appropriate section.

#### 4.3.1 Bus Control Operations

The function of bus control transfer was added to bus control operations as explained in the next section.

##### 4.3.1.1 Bus Control Transfer

Bus control transfer in the nonstationary master system is accomplished by using the dynamic bus control mode code provided in MIL-STD-1553B. This mode code, plus other messages as described in the control procedures, is contained in a special purpose bus instruction list. By using a separate bus instruction list rather than putting the bus control transfer commands at the end of the normal synchronous instruction list it allows for status polling and scheduled low priority asynchronous messages to occur after the synchronous instruction list and before the bus control transfer.

In order to recover from bus control transfer failures, the bus control management function, resident in the primary master, must be notified of the upcoming change. The bus control transfer protocol, therefore, includes a message to bus control management.

##### 4.3.1.2 Minor Cycle Synchronization

For the nonstationary master system, the primary master is in control of the system. Therefore, it is the primary master that initiates the minor cycle synchronization on the NSMITS bus. The changes in bus synchronization are explained in the next paragraph.

#### 4.3.1.2.1 Bus Synchronization

Bus synchronization in the NSMITS system is essentially the same as in the SMITS system. The difference is that in the NSMITS system the new minor cycle is not started until the expiration of the minor cycle clock in the primary master and the completion of the round robin sequence of bus control transfer for the current minor cycle. Secondary masters do not maintain a minor cycle clock.

#### 4.3.1.2.2 Local Executive Minor Cycle Setup

No change.

#### 4.3.1.2.3 Master Executive Minor Cycle Setup

The secondary masters do not perform resetting of the minor cycle clock, but do set up the next sequence of transmission lists.

#### 4.3.1.3 Synchronous Bus Message Operation

In a SMITS system, the new minor cycle can start after the completion of the synchronous instruction list. In the NSMITS system, however, bus control transfer or the start of a new minor cycle can occur after the completion of the synchronous instruction list depending on whether the processor is a primary or secondary master.

##### 4.3.1.3.1 Processor/BIU DMA Sequence (Master Mode)

In the NSMITS system, the primary master at the start of a minor cycle and the secondary masters when they gain control of the bus, perform this sequence.

##### 4.3.1.3.2 Processor/BIU DMA Sequence (Remote Mode)

No change.

#### 4.3.1.4 Asynchronous Bus Message Operation

No change.

#### 4.3.1.4.1 Interprocessor Asynchronous Messages

No change.

#### 4.3.1.4.2 Critically Timed Asynchronous Message Operation

To provide simplicity and low overhead, each master on the NSMITS bus will handle critically timed messages for devices in its sphere of control when it is master.

#### 4.3.1.4.4 Status Polling

Provisions have been made in the NSMITS for each master to do a round of status polling each time that it is master. To obtain the maximum multimission capability possible, each master is allowed to do status polling instead of only the primary master. Because a master may communicate with RT's that no other master talks with, each master will do a status poll of only those RT's in its sphere of control.

#### 4.3.2 Mode Command Operations

The dynamic bus control mode code has been added to the mode command operations of the NSMITS system to allow for bus control transfer.

#### 4.3.3 Error Failure/Management

No change.

##### 4.3.3.1 Message Retry Classes

No change.

##### 4.3.3.2 Interrupt Processing (Master and Remote)

No change.

#### 4.3.3.3 Status Word Analysis

No change.

#### 4.3.3.4 BIT Word Request and Analysis

No change.

#### 4.3.3.5 Terminal Failure Analysis

No change.

#### 4.3.3.6 Core Elements

No change.

#### 4.3.3.7 "FATAL ERROR" Management

No change.

#### 4.3.4 RT Serial/Digital Channel Operation

No change.

#### 4.3.5 Application Executive Services

No change.

#### 4.3.6 Mission Application Tasks

No change.

#### 4.3.7 Configuration Management

No change.

#### 4.3.8 Bus Control Management

A new executive function, bus control management, has been defined for the NSMITS. Bus control management resides in the primary master and monitor processors as part of the system control executive. Its function is to monitor the round robin sequence of bus control transfer and to determine the processor at fault if a bus control transfer fails. Bus control management then reports the failure to configuration management which can perform a warm start on the bus to eliminate the failed processor from the bus control transfer sequence via coordination with affected secondary masters.

The reason that bus control management is not part of the configuration function is that bus control management is required only to support the primary master function, whereas the configuration management function is required in secondary masters for devices in their sphere of control.

#### 4.3.9 Monitor Management

No change.

#### 4.4 BACKUP OPERATION

No change.

#### 4.5 SYSTEM RECONFIGURATION OPERATION

No change.

APPENDIX C  
RATIONALE FOR CONTENTION MULTIPLE ACCESS INFORMATION TRANSFER  
SYSTEM CONTROL PROCEDURES

The system control procedures for CMAITS are substantially different from DAIS. This discussion will not be directed as much toward changes to DAIS control procedures as to specific problems encountered in the design of the contention control procedures. This appendix assumes that the reader has available both the contention ITS description of Interim Report #1, Volume 2, Appendix C and the Contention Multiple Access ITS Control Procedures, Volume 3 of CDRL #18.

Contention Protocol Definition

The protocol of the contention system was defined to be substantially different from MIL-STD-1553 because of two factors: (1) The messages are addressed by content rather than by device number, and (2) the messages are broadcast so that any combination of devices may receive any particular message. The Manchester bi-phase form of data encoding was chosen to be the same as 1553B, but the format of the command words were different (see Interim Report #1, Appendix C for the detailed discussion). The address and subaddress fields were merged to allow a 10 bit message identifier field. MIL-STD-1553 was considered as a candidate using the broadcast mode and using the message identifier (address) as the first word in the message. This method was rejected for the CMAITS because the protocol is still oriented around command-response, and additionally would require an additional overhead word on each transmission. However, if a contention ITS were to be integrated into a bus system with a command-response set of equipment, this protocol would be a likely candidate.

One concern of the broadcast system was that all of the destination devices would not receive a broadcast message. It seems likely that a BIU, in a contention ITS may transmit a message which cannot be interpreted by intended user terminals either due to waveform errors or data validity check errors. However, the error may not be detected by the transmitting BIU. The Manchester waveform may be valid near the transmitting BIU, but not far along the bus. A means has provided to notify the transmitting BIU that some terminal, which may have needed the data, could not decode the message properly. Since the predominant transmission mode in the contention ITS is broadcast, no particular terminal can be identified as the intended destination.

A selected BIU could be designated as "bus monitor BIU" and would detect bus errors. The bus monitor might transmit its error notice word (with a word sync waveform like a Message/Request ID word) immediately upon detecting a quiescent bus at the end of the bad message. It should avoid overlapping the bad message. (The bus terminals must all detect a quiescent bus for a somewhat longer interval before initiating their contention process, in order to allow enough time for the bus monitor to detect a quiescent bus, complete the error detection function and begin transmitting the error notice word.

The bus monitor may be nearer, along the bus, to the transmitter than some other terminal or than some bus fault. Therefore, it seems that all, or at least several, bus terminals must be allowed to act as bus monitors. In this arrangement it seems likely that two or more bus monitors may produce a collision of

error notice words. The error notice words would probably thereby be made non-decodable. However, they still serve their purpose. Transmitting BIU's should recognize any bus activity within a particular interval after their end of transmission as an error notice word. If an error notice word should collide with a message ID word which follows a request ID word in a sync type transaction, the requesting BIU may perform error recovery as if the message ID word were defective without regard to the cause of the error. However, the BIU which initiated the request should not assume that the error is in the interrogated BIU.

When a normal collision occurs, the terminals which are not transmitting will detect Manchester waveform errors. These terminals cannot distinguish collisions from other waveform errors. In this case all listening terminals will transmit error notice words.

The time during which error notice words exclusively occupy the bus should be minimized. Perhaps a shorter transmission than a 20-bit-time word will suffice for an error notice. It is also desirable to avoid waste of bus time completing a message in which a defect has been detected. It may be satisfactory to initiate an error notice word transmission anytime after the initial collision detection interval. The BIU's which initiate messages should perform continual waveform analysis through the duration of the message, and should treat any abnormality as an error notice.

It remains desirable for the transmitting BIU's to distinguish between normal collisions at the beginning of the message and later errors for diagnostic and error recovery purposes. If collisions can be detected by the transmitting BIU's in a shorter interval than a complete word transmission time, then the colliding transmissions can be terminated before complete word transmission. Monitoring BIU's should ignore collision events which are represented by transmissions of less than one word length.

The collision detection mechanism in the BIU is not well defined, although some engineering work was done to determine its feasibility using Manchester phase protocol. Detection of collisions may depend partly on different bit patterns in message ID words which can be transmitted by different BIU's. If there are Message ID words differing only in the last information bit, which can be transmitted by different BIU's, then detection of collisions will be delayed until the end of the first word. If the detection mechanism depends on bit differences, then it would be desirable to concentrate the differences in message ID words in the leading part of the word. This discrimination could be accomplished by assigning a large bias number different for each BIU in the Message ID field.

The control procedures also provide a discussion on the allocation of message ID's among processing elements to compensate for BIU collision detection circuitry which might not be able to detect a collision quickly during early development of collision circuitry. The ID's provide adequate bit differences during the initial transmission of the first word of a message to aid in the detection of multiple messages simultaneously transmitted.

#### Cyclic Time Slot Bus Access Protocol Alternative

One attractive alternative bus protocol which was examined and rejected because

it was not quite as general as total contention in the context of multi-mission applications was that of variable time slots. This access protocol avoids message collisions and the necessity of collision detection circuitry in the terminals. During the bus quiet period each terminal is assigned a definite and limited transmission opportunity or time slot. The sequence of time slots is repeated continually in regular order. When a terminal uses its time slot to begin transmission, the passage of time slots is suspended until the message period is complete.

Time-slot timing is maintained independently in each terminal, and all are resynchronized by each message on the bus. The zero voltage bus signal crossing in the middle of the last word in the synchronization wave form is used by all terminals as the time slot reference signal. The time slot deviations among the terminals are the sum of the following time increments:

- a. The maximum bus signal propagation delay between two terminals (usually the two terminals with the longest signal path between them).
- b. The maximum time skew of the synchronization reference at two terminals due to bus signal distortion and noise.
- c. The maximum difference between two terminals' synchronization reference acquisition time, and
- d. The maximum relative drift between two terminals' time slot time bases during the bus quiet period.

Bus messages must occur at some maximum interval which is established to limit the drift component of the time slot deviation. If system communications are not frequent enough, then a special message must be provided. Figure A-1 shows an example of the message and time slot timing.



$M_{H1}$  Data Message Transmitted at Normal Priority by Terminal "1"  
 $M_{H1}$  Data Message Transmitted at High Priority by Terminal "1"  
 $T_{H1}$  High Priority Access Time Slot for Designated Terminal "1"  
 $T_R$  Response Data Message Access Time Slot Following Request Message  
 $RM$  Request Message  
 $MR$  Response Data Message  
 $T_{N1}$  Normal Priority Access Time Slot for Terminal "1"  
 $G$  After Message Gap Interval Reserved for Error Notice Signal Burst  
 $*$  Response Message Omitted, Resume High Priority Time Slot Cycle

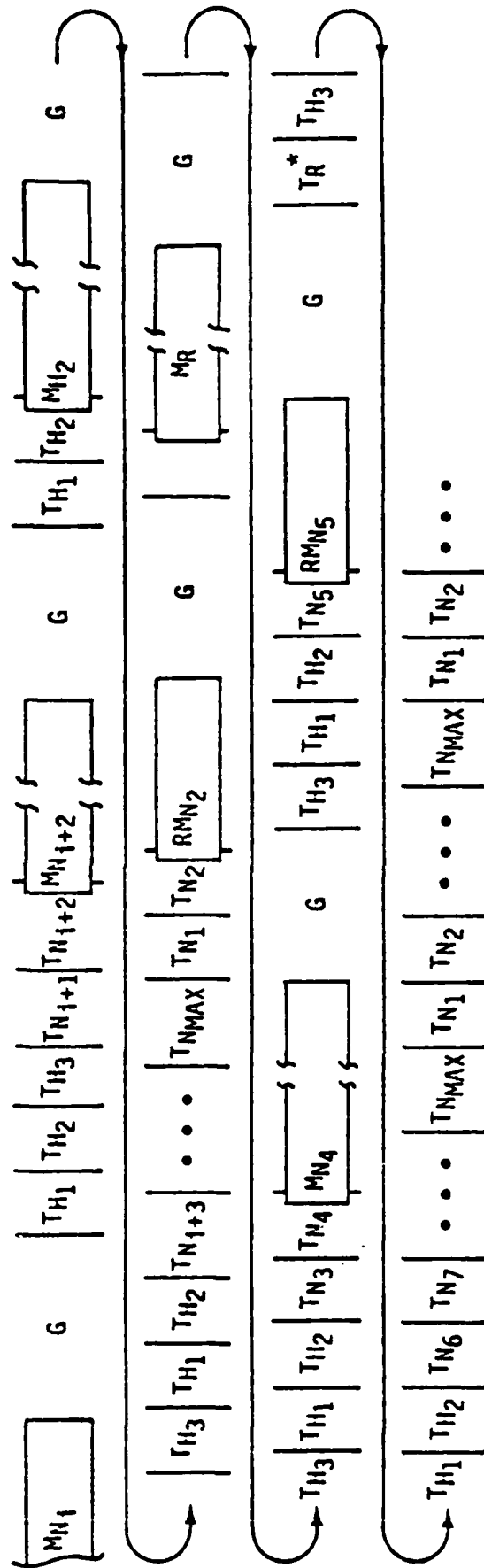


Figure C-1. Time Slot Allocation

The time slot passage resumes TBD microseconds after the synchronization reference event in the immediately preceding message. This delay includes the remainder of the final word of the message and a fixed period when the bus is normally quiet after any message, which allows for an error notice signal. The first time slot after any request message word (request bit set in message ID word) is reserved for the data message response to the request. After a data message, the response time slot is omitted.

A short cycle of time slots is provided for transmission of high priority messages by a limited number of designated terminals. If a high priority time slot is used, the high priority cycle resumes from the time slot of suspension when the message interval is concluded. A complete cycle of high priority message time slots must follow any message before the cycle of normal priority time slots resumes. The cycle of normal priority time slots resumes from the time slot of its suspension.

Each transmission must be timed so that it is detected in the same numbered time slot by all terminals. Therefore, the transmission beginning must be delayed from the terminal's time slot beginning by an interval at least equal to the maximum time slot deviation.

The time slot duration is determined by the sum of the following increments:

- e. The maximum time slot starting deviation among the terminals (sum of a, b, c, d),
- f. The transmission starting delay from the time slot beginning,
- g. The maximum bus propagation delay for the new message,
- h. The maximum sampling and processing time for any terminal to determine that transmission is in process and suspend the passage of time slots, and
- j. The time slot integrity margin.

While this protocol was initially rejected, it still appears to be an excellent alternative protocol if a totally asynchronous access protocol is not acceptable to a user because of a (perceived or real) requirement for synchronized access to the bus.

#### Normal Operation

The normal operations of the CMAITS are expected to run with a moderate to low number of collisions. The analyses performed in appendices C and D of the First Interim Report showed that for what is expected for a "normal" set of avionics traffic, the contention scheme should work well. Two contingencies have been provided if the system starts to overload. If the overload is a temporary collision overload, the entire transmission sequence is designed to slow down and has been shown (ref. CACM, July 1976 pp 395-403) that it will not collapse from too many desired communications. The protection mechanism is in the way that the BIU computes its random waits following a collision. If the bus overload condition continually occurs, then the transmission medium should be increased. Fiber optics buses easily can accommodate 10 megabit transmission rates, which could alleviate any foreseeable overload condition.

The hierarchical influence is minimal on the contention scheme because of the independence designed into it. If two buses attach to a single PE, then we assume that the PE is a part of at most a single minor cycle sequence external to its internal minor cycle operation. In actuality we expect that most devices will be operating on their own minor cycles independently from the others, especially at a node in the hierarchy.

The normal operations of the bus control include setting queue entries for the BIU to transmit. There are four queues, for four distinct priority levels which were identified: (1) responses to message requests from other PE's, (2) trigger (high priority asynchronous) messages, (3) synchronous messages, and (4) asynchronous messages. Because of the multiple priority levels, we felt that the simplest approach to message management would be to add message transmission queues and to garbage collect as appropriate.

The mass memory is used in normal operation to record mission and error data and to retrieve data such as navigation aids. The mass memory is attached to a single bus, and queues all requests from the independently operating PE's. Because of the asynchronous behavior of the PE's, queueing of requests and sequential responses seemed to be the most appropriate. The status of each request to mass memory is however returned to the individual requestor prior to any mass memory activity.

#### Abnormal Operations

Message errors are expected to be the most frequent type of an error. The broadcast method precludes error notification except with the use of error monitors, which are additional capabilities included in some BIU's, as was discussed under normal operations. The grave concern about the use of such an option is that an error monitor's two receivers could fail in such a way that every message transmission could be determined to be invalid. The consequence of such a failure is that the entire message traffic of the bus would be effectively halted. The best option would be to require that the bus monitor be used to control the configuration management of some of the devices and would be collocated with a sphere controller.

The sphere controller is similar to the DAIS master executive in its functions of minor cycle control and configuration control. If specific devices are interrelated according to function, then the sphere controller assumes control for that combination of devices, and performs its control based upon an initial polling of functions available during startup, so that alternative configurations of functioning can be computed.

The sphere controller was added to the totally asynchronously operating ITS as an option because some operations may be defined to require a synchronizing activity. Similarly some level of configuration management may be required over the bus. However, totally redundant devices could be integrated into the system structure by operating concurrently, and each receiving device sorting out which data is to be selected. This approach is the mode in which most avionics devices currently operate on commercial and military aircraft. Hierarchical bus architectures are used in these systems to totally separate redundant devices into pilot and copilot sides, each sharing common (single) sensors and functions. The goal in these configurations is to maintain separation and protection of one set

of avionics devices from another set of devices. Triple redundant and single devices must be shared by the two separate sets of devices. This concept of independence of function is a motivation to have a CMAITS which has neither configuration management nor synchronization functions (i.e. no sphere controller functions).